Guida per gli sviluppatori

Amazon API Gateway

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon API Gateway: Guida per gli sviluppatori

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

I marchi e il trade dress di Amazon non possono essere utilizzati in relazione ad alcun prodotto o servizio che non sia di Amazon, in alcun modo che possa causare confusione tra i clienti, né in alcun modo che possa denigrare o screditare Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà delle rispettive aziende, che possono o meno essere associate, collegate o sponsorizzate da Amazon.

Table of Contents

Cos'è Amazon API Gateway? ... 1
Architettura di API Gateway ... 2
Caratteristiche di Gateway API ... 2
APICasi d'uso del gateway ... 3
Usa API Gateway per creare REST APIs ... 4
Usa API Gateway per creare HTTP APIs ... 4
Usa API Gateway per creare WebSocket APIs ... 5
Chi utilizza API Gateway? ... 6
Accesso a API Gateway ... 6
Parte dell'infrastruttura serverless AWS ... 7
Come iniziare a usare Amazon API Gateway ... 7
APIConcetti di gateway ... 8
Scegli tra REST APIs e HTTP APIs ... 13
... 13
Tipo di endpoint ... 13
Sicurezza ... 14
Autorizzazione ... 14
APIgestione ... 15
Sviluppo ... 16
Monitoraggio ... 17
Integrazioni ... 17
Inizia a usare la REST API console ... 18
Fase 1: creazione di una funzione Lambda ... 19
Fase 2: Creare un REST API ... 20
Fase 3: Creazione di un'integrazione proxy Lambda ... 21
Fase 4: Implementa il API ... 21
Passaggio 5: richiama il API ... 21
(Opzionale) Fase 6: pulizia ... 22
Prerequisiti ... 24
Registrati per un Account AWS ... 24
Crea un utente con accesso amministrativo ... 24
Inizia a usare ... 26
Fase 1: creazione di una funzione Lambda ... 27
Fase 2: Creare un HTTP API ... 27

Fase 3: Metti alla prova il tuo API ... 28
(Opzionale) Fase 4: pulizia ... 29
Passaggi successivi ... 30
Tutorial e workshop ... 32
RESTAPItutorial ... 33
Scegli un tutorial sull'integrazione con Lambda ... 33
Tutorial: creazione di un'API REST mediante l'importazione di un esempio ... 57
Scegli un tutorial sull'integrazione HTTP ... 66
Tutorial: crea un'API REST con un'integrazione privata ... 81
Tutorial: crea un'API REST con un' AWS integrazione ... 84
Tutorial: crea una calcolatrice con tre integrazioni REST API ... 90
Tutorial: creare un REST API proxy come Amazon S3 ... 120
Tutorial: creazione di un'API REST come un proxy Amazon Kinesis ... 165
Tutorial: crea un'API REST utilizzando AWS SDK o AWS CLI ... 211
Tutorial: crea un account privato REST API ... 244
Tutorial sull'API HTTP ... 251
Tutorial: Crea un'API HTTP CRUD con Lambda e DynamoDB ... 251
Tutorial: crea un HTTP API account con un'integrazione privata con Amazon ECS ... 263
WebSocket Tutorial sulle API ... 270
Tutorial: crea un'app di chat WebSocket ... 271
Tutorial: Crea un WebSocket API file con un' AWS integrazione ... 276
API REST di API Gateway ... 291
Sviluppare ... 291
Tipi di endpoint API Gateway ... 292
Metodi ... 297
Controllo accessi ... 316
Integrazioni ... 402
Convalida delle richieste ... 471
Trasformazioni dei dati ... 501
Risposte del gateway ... 574
CORS ... 586
Tipi di supporti binari ... 601
Invoke ... 633
Apri API ... 665
Pubblicare ... 679
Implementa REST APIs ... 680

Nomi di dominio personalizzati ... 725
Ottimizzazione ... 768
Impostazioni cache ... 769
Codifica del contenuto ... 779
Distribuzione ... 785
Piani di utilizzo ... 785
APIdocumentazione ... 813
SDKgenerazione ... 880
Vendi le tue API come SaaS ... 907
Protezione ... 911
Mutuo TLS ... 912
Certificati del client ... 919
AWS WAF ... 960
Throttling ... 963
Privato REST APIs ... 966
Monitoraggio ... 982
CloudWatch metriche ... 983
CloudWatch registri ... 993
Firehose ... 999
X-Ray ... 1000
API HTTP API Gateway ... 1015
Sviluppare ... 1015
Creazione di un'API HTTP ... 1016
Route ... 1017
Controllo accessi ... 1020
Integrazioni ... 1039
CORS ... 1061
Mappatura dei parametri ... 1063
OpenAPI ... 1071
Pubblicare ... 1080
Stage ... 1081
Politica di sicurezza per HTTP APIs in API Gateway ... 1084
Nomi di dominio personalizzati ... 1086
Protezione ... 1094
Throttling ... 1094
Mutuo TLS ... 1095

Monitoraggio ... 1102
Metriche ... 1103
Registrazione ... 1105
Risoluzione dei problemi ... 1116
Integrazioni Lambda ... 1116
Provider di autorizzazioni JWT ... 1119
APIGateway WebSocket APIs ... 1121
Panoramica di WebSocket APIs ... 1121
Usa i percorsi per un WebSocket API ... 1122
Invia dati alle app client connesse ... 1122
Gestisci gli utenti connessi e le app client ... 1123
Richiama la tua integrazione con il backend ... 1126
WebSocket espressioni di selezione ... 1131
Sviluppare ... 1141
Creazione e configurazione ... 1141
Route ... 1143
Controllo accessi ... 1152
Integrazioni ... 1160
Convalida delle richieste ... 1170
Trasformazioni dei dati ... 1173
Tipi di supporti binari ... 1186
Invoke ... 1186
Pubblicare ... 1190
Stage ... 1190
Implementa un'API WebSocket ... 1193
Politica di sicurezza per WebSocket APIs in API Gateway ... 1196
Nomi di dominio personalizzati ... 1198
Protezione ... 1204
Limitazione a livello di account per regione ... 1204
Throttling a livello di instradamento ... 1205
Monitoraggio ... 1205
Metriche ... 1206
Registrazione ... 1208
ARN di API Gateway ... 1217
API HTTP e risorse WebSocket API ... 1217
Risorse API REST ... 1220

`execute-api`(API HTTP, WebSocket API e API REST) 1225
APIEstensioni aperte 1226
`x-amazon-apigateway-any-method` 1227
x-amazon-apigateway-any-method: esempi 1228
x-amazon-apigateway-cors 1229
x-amazon-apigateway-cors esempio 1229
`x-amazon-apigateway-api-key-source` 1230
x-amazon-apigateway-apiesempio di -key-source 1231
`x-amazon-apigateway-auth` 1232
x-amazon-apigateway-auth esempio 1232
`x-amazon-apigateway-authorizer` 1233
x-amazon-apigateway-authorizer esempi per REST APIs 1236
x-amazon-apigateway-authorizer esempi per HTTP APIs 1240
`x-amazon-apigateway-authtype` 1242
x-amazon-apigateway-authtype esempio 1242
Consulta anche 1244
x-amazon-apigateway-binary-tipo di supporto 1244
x-amazon-apigateway-binaryesempio di -media-types 1245
x-amazon-apigateway-documentation 1245
x-amazon-apigateway-documentation esempio 1245
x-amazon-apigateway-endpoint-configurazione 1246
x-amazon-apigateway-endpoint-esempi di configurazione 1247
x-amazon-apigateway-gateway-risposte 1247
x-amazon-apigateway-gateway-esempio di risposte 1247
x-amazon-apigateway-gateway-risposte. gatewayResponse 1248
x-amazon-apigateway-gateway-risposte. gatewayResponse esempio 1249
x-amazon-apigateway-gateway-risposte. responseParameters 1249
x-amazon-apigateway-gateway-risposte. responseParameters esempio 1250
x-amazon-apigateway-gateway-risposte. responseTemplates 1250
x-amazon-apigateway-gateway-risposte. responseTemplates esempio 1251
x-amazon-apigateway-importexport-versione 1251
x-amazon-apigateway-importexport-esempio di versione 1251
`x-amazon-apigateway-integration` 1252
x-amazon-apigateway-integration esempi 1257
x-amazon-apigateway-integrations 1259
x-amazon-apigateway-integrations esempio 1259

x-amazon-apigateway-integration.requestTemplates 1261
x-amazon-apigateway-integration. requestTemplates esempio 1261
x-amazon-apigateway-integration.requestParameters 1262
`x-amazon-apigateway-integration.requestParametersEsempio` 1263
x-amazon-apigateway-integration.risposte 1264
`x-amazon-apigateway-integration.responsesEsempio` 1265
x-amazon-apigateway-integration.risposta 1266
`x-amazon-apigateway-integration.responseEsempio` 1267
x-amazon-apigateway-integration.responseTemplates 1267
x-amazon-apigateway-integration. responseTemplate esempio 1268
x-amazon-apigateway-integration.responseParameters 1268
`x-amazon-apigateway-integration.responseParameters` Esempio 1269
x-amazon-apigateway-integration.tlsConfig 1269
x-amazon-apigateway-integration. tlsConfig esempi 1271
x-amazon-apigateway-minimum-dimensione di compressione 1272
x-amazon-apigateway-minimum-compression-size, esempio 1272
x-amazon-apigateway-policy 1272
`x-amazon-apigateway-policyEsempio` 1273
x-amazon-apigateway-request-validatore 1273
`x-amazon-apigateway-request-validator` Esempio 1274
x-amazon-apigateway-request-validatori 1274
`x-amazon-apigateway-request-validatorsEsempio` 1275
x-amazon-apigateway-request-validatori. requestValidator 1276
`x-amazon-apigateway-request-validators.requestValidatorEsempio` 1276
x-amazon-apigateway-tag-valore 1277
`x-amazon-apigateway-tag-valueEsempio` 1277
Sicurezza 1278
Protezione dei dati 1279
Crittografia dei dati 1280
Riservatezza del traffico Internet 1281
Gestione dell'identità e degli accessi 1281
Destinatari 1281
Autenticazione con identità 1282
Gestione dell'accesso con policy 1285
Come funziona Amazon API Gateway con IAM 1288
Esempi di policy basate su identità 1293

Esempi di policy basate su risorse ... 1302
Risoluzione dei problemi ... 1302
Utilizzo di ruoli collegati ai servizi ... 1304
Logging e monitoraggio ... 1309
Lavorare con CloudTrail ... 1310
Lavorare con AWS Config ... 1313
Convalida della conformità ... 1317
Resilienza ... 1318
Sicurezza dell'infrastruttura ... 1319
Analisi della configurazione e delle vulnerabilità ... 1320
Best practice ... 1320
Applicazione di tag ... 1322
Risorse API Gateway a cui è possibile applicare tag ... 1323
Eredità dei tag nell'API di Amazon API Gateway V1 ... 1324
Limitazioni applicate ai tag e convenzioni di utilizzo ... 1325
Controllo dell'accesso basato sugli attributi ... 1326
Limitare le operazioni in base ai tag delle risorse ... 1326
Consentire operazioni in base ai tag delle risorse ... 1327
Negare operazioni di assegnazione di tag ... 1328
Consentire operazioni di assegnazione di tag ... 1329
Riferimenti API ... 1331
Quote e note importanti ... 1332
APIQuote a livello di account Gateway, per regione ... 1332
HTTPAPIquote ... 1333
... 1333
APIQuote del gateway per la configurazione e l'esecuzione di un WebSocket API ... 1336
APIQuote del gateway per la configurazione e l'esecuzione di un REST API ... 1338
APIQuote gateway per la creazione, l'implementazione e la gestione di un API ... 1343
Note importanti ... 1345
Note importanti per REST APIs HTTPAPIs, e WebSocket APIs ... 1345
Note importanti su Amazon API Gateway per HTTP APIs ... 1345
Note importanti per REST APIs e WebSocket APIs ... 1346
Note importanti per WebSocket APIs ... 1346
Note importanti per REST APIs ... 1346
Cronologia dei documenti ... 1353
Aggiornamenti precedenti ... 1365

..... mccclxxvi

Cos'è Amazon API Gateway?

Amazon API Gateway è un AWS servizio per la creazione, la pubblicazione, la manutenzione, il monitoraggio e la protezione RESTHTTP, WebSocket APIs su qualsiasi scala. APIgli sviluppatori possono creare APIs quell'accesso AWS o altri servizi web, oltre ai dati archiviati nel AWS cloud. In qualità di API sviluppatore API Gateway, puoi creare APIs applicazioni client da utilizzare nelle tue applicazioni client. Oppure puoi renderlo APIs disponibile agli sviluppatori di app di terze parti. Per ulteriori informazioni, consulta the section called “Chi utilizza API Gateway?”.

APIGateway crea RESTful APIs che:

- Sono HTTP basati.
- Consentono la comunicazione client-server stateless.
- Implementa HTTP metodi standard come GETPOST,PUT,PATCH, eDELETE.

Per ulteriori informazioni su API Gateway REST APIs and HTTP APIsthe section called “Scegli tra REST APIs e HTTP APIs ”, vedereAPI HTTP API Gateway,the section called “Usa API Gateway per creare REST APIs”, ethe section called “Sviluppare”.

APIGateway crea WebSocket APIs che:

- Aderisci al WebSocketprotocollo, che consente una comunicazione full-duplex basata sullo stato tra client e server.
- Instradano i messaggi in ingresso e sono basate sul contenuto dei messaggi.

Per ulteriori informazioni su API Gateway, vedere e. WebSocket APIs the section called “Usa API Gateway per creare WebSocket APIs” the section called “Panoramica di WebSocket APIs”

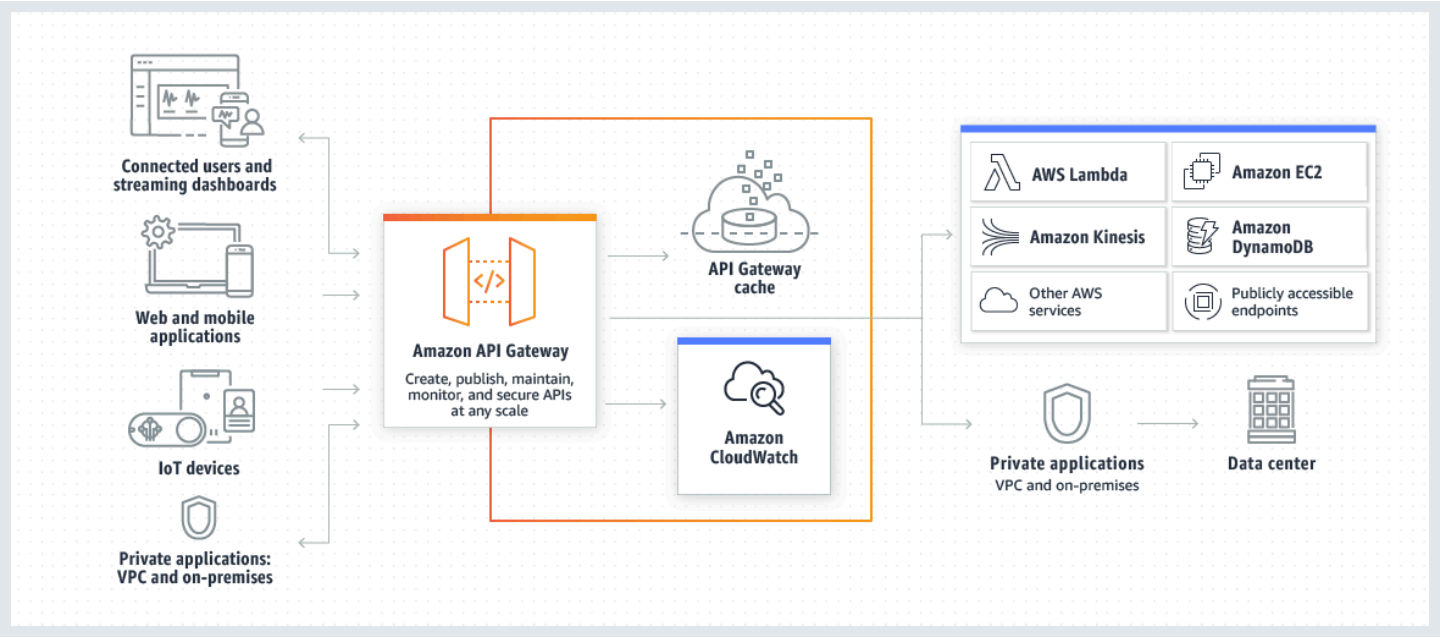
Argomenti

- Architettura di API Gateway
- Caratteristiche di Gateway API
- APICasi d'uso del gateway
- Accesso a API Gateway
- Parte dell'infrastruttura serverless AWS
- Come iniziare a usare Amazon API Gateway

- Concetti di Amazon API Gateway
- Scegli tra REST APIs e HTTP APIs
- Inizia a usare la REST API console

Architettura di API Gateway

Il diagramma seguente mostra l'architettura API Gateway.



Questo diagramma illustra come lo APIs sviluppo di Amazon API Gateway offra a te o ai tuoi clienti sviluppatori un'esperienza di sviluppo integrata e coerente per la creazione di applicazioni AWS serverless. APIGateway gestisce tutte le attività necessarie per accettare ed elaborare fino a centinaia di migliaia di chiamate simultanee. API Queste attività includono la gestione del traffico, il controllo delle autorizzazioni e degli accessi, il monitoraggio e la gestione delle API versioni.

APIIl gateway funge da «porta d'ingresso» per consentire alle applicazioni di accedere ai dati, alla logica di business o alle funzionalità dei tuoi servizi di backend, come carichi di lavoro in esecuzione su Amazon Elastic Compute Cloud (AmazonEC2), codice in esecuzione su AWS Lambda qualsiasi applicazione Web o applicazioni di comunicazione in tempo reale.

Caratteristiche di Gateway API

Amazon API Gateway offre funzionalità come le seguenti:

- Support per stateful (WebSocket) e stateless (HTTPand). RESTAPIs
- Meccanismi di autenticazione potenti e flessibili, come AWS Identity and Access Management policy, funzioni di autorizzazione Lambda e pool di utenti di Amazon Cognito.
- Distribuzioni di versioni di Canary per l'implementazione sicura di modifiche.
- CloudTrailregistrazione e monitoraggio dell'utilizzo e delle modifiche. API API
- CloudWatch registrazione degli accessi e registrazione delle esecuzioni, inclusa la possibilità di impostare allarmi. Per ulteriori informazioni, consulta the section called “CloudWatch metriche” e the section called “Metriche”.
- Possibilità di utilizzare AWS CloudFormation modelli per consentire la creazione. API Per ulteriori informazioni, consulta Amazon API Gateway Resource Types Reference e Amazon API Gateway V2 Resource Types Reference.
- Supporto per nomi di dominio personalizzati.
- Integrazione con AWS WAFper APIs proteggerti dagli exploit web più comuni.
- Integrazione con AWS X-Ray per identificare e valutare le prestazioni in termini di latenza.

Per un elenco completo delle versioni delle funzionalità di API Gateway, consultaCronologia dei documenti.

APICasi d'uso del gateway

La seguente sezione sui casi d'uso presenta una panoramica dei diversi tipi di API Gateway APIs e dei diversi tipi di sviluppatori che utilizzano API Gateway. Per informazioni più dettagliate sulla differenza tra REST APIs e HTTPAPIs, vederethe section called “Scegli tra REST APIs e HTTP APIs”.

Argomenti

- Usa API Gateway per creare REST APIs
- Usa API Gateway per creare HTTP APIs
- Usa API Gateway per creare WebSocket APIs
- Chi utilizza API Gateway?

Usa API Gateway per creare REST APIs

Un API gateway REST API è composto da risorse e metodi. Una risorsa è un'entità logica a cui un'app può accedere tramite un percorso della risorsa. Un metodo corrisponde a una REST API richiesta inviata dall'utente dell'utente API e alla risposta restituita all'utente.

Ad esempio, `/incomes` potrebbe essere il percorso di una risorsa che rappresenta il reddito dell'utente dell'app. Una risorsa può avere una o più operazioni definite da HTTP verbi appropriati come GETPOST, PUTPATCH, eDELETE. Una combinazione di un percorso di risorsa e un'operazione identifica un metodo di. API Ad esempio, un metodo `POST /incomes` potrebbe aggiungere il reddito guadagnato dall'intermediario, mentre il metodo `GET /expenses` potrebbe eseguire una query relativa alle spese indicate sostenute dall'intermediario.

Non è necessario che l'app sappia dove i dati vengono archiviati e prelevati sul back-end. In API Gateway RESTAPIs, il frontend è incapsulato dalle richieste di metodo e dalle risposte dei metodi. Le API interfacce con il backend mediante richieste di integrazione e risposte di integrazione.

Ad esempio, con DynamoDB come backend, API lo sviluppatore imposta la richiesta di integrazione per inoltrare la richiesta del metodo in entrata al backend scelto. La configurazione include le specifiche di un'azione DynamoDB appropriata, il ruolo e le politiche IAM richiesti e la trasformazione dei dati di input richiesta. Il backend restituisce il risultato a API Gateway come risposta di integrazione.

Per indirizzare la risposta di integrazione a una risposta del metodo appropriata (di un determinato codice di HTTP stato) al client, puoi configurare la risposta di integrazione per mappare i parametri di risposta richiesti dall'integrazione al metodo. Quindi puoi trasformare il formato dei dati di output del back-end nel formato del front-end, se necessario. APIGateway consente di definire uno schema o un modello per il payload per facilitare la configurazione del modello di body mapping.

APIGateway offre funzionalità di REST API gestione come le seguenti:

- Support per la generazione SDKs e la creazione di API documentazione utilizzando le estensioni API Gateway to Open API
- Limitazione delle richieste HTTP

Usa API Gateway per creare HTTP APIs

HTTPAPIsconsentono di creare RESTful APIs con una latenza inferiore e costi inferiori rispetto a RESTAPIs.

È possibile utilizzare HTTP APIs per inviare richieste a AWS Lambda funzioni o a qualsiasi endpoint instradabile HTTP pubblicamente.

Ad esempio, puoi creare un file HTTP API che si integri con una funzione Lambda sul backend. Quando un client chiama il tuoAPI, API Gateway invia la richiesta alla funzione Lambda e restituisce la risposta della funzione al client.

HTTPAPIssupporta l'autorizzazione OpenID Connect e OAuth2.0. Sono dotati di supporto integrato per la condivisione di risorse tra origini diverse (CORS) e le distribuzioni automatiche.

Per ulteriori informazioni, consulta the section called “Scegli tra REST APIs e HTTP APIs ”.

Usa API Gateway per creare WebSocket APIs

In a WebSocket API, il client e il server possono scambiarsi messaggi in qualsiasi momento. I server di back-end possono eseguire facilmente il push dei dati agli utenti e ai dispositivi connessi, evitando la necessità di implementare complessi meccanismi di polling.

Ad esempio, è possibile creare un'applicazione serverless utilizzando un API gateway WebSocket API e AWS Lambda inviare e ricevere messaggi da e verso singoli utenti o gruppi di utenti in una chat room. Oppure puoi richiamare servizi di backend come AWS Lambda Amazon Kinesis o HTTP un endpoint basato sul contenuto dei messaggi.

Puoi utilizzare API Gateway WebSocket APIs per creare applicazioni di comunicazione sicure e in tempo reale senza dover fornire o gestire server per gestire connessioni o scambi di dati su larga scala. I casi d'uso di riferimento includono applicazioni in tempo reale come le seguenti:

- Applicazioni di chat
- Pannelli di controllo in tempo reale, ad esempio ticker azionari
- Avvisi e notifiche in tempo reale

APIGateway offre funzionalità di WebSocket API gestione come le seguenti:

- Monitoraggio e throttling di connessioni e messaggi
- Viene utilizzato AWS X-Ray per tracciare i messaggi mentre viaggiano attraverso i APIs servizi di backend
- Facile integrazione conHTTP/endpoints HTTPS

Chi utilizza API Gateway?

Esistono due tipi di sviluppatori che utilizzano API Gateway: API sviluppatori e sviluppatori di app.

Uno API sviluppatore crea e distribuisce un file API per abilitare le funzionalità richieste in API Gateway. Lo API sviluppatore deve essere un utente dell' AWS account proprietario diAPI.

Uno sviluppatore di app crea un'applicazione funzionante per chiamare AWS i servizi richiamando WebSocket o REST API creata da uno API sviluppatore in API Gateway.

Lo sviluppatore dell'app è il cliente dello API sviluppatore. Lo sviluppatore dell'app non deve disporre di un AWS account, a condizione che non richieda IAM autorizzazioni o supporti l'APIautorizzazione degli utenti tramite provider di identità federati di terze parti supportati dalla federazione delle identità dei pool di utenti di Amazon Cognito. Tra i provider di identità figurano Amazon, pool di utenti Amazon Cognito, Facebook e Google.

Creazione e gestione di un gateway API API

Uno API sviluppatore utilizza il componente del servizio API Gateway per la API gestione`apigateway`, denominato, per creare, configurare e distribuire unAPI.

In qualità di API sviluppatore, puoi creare e gestire un file API utilizzando la console API Gateway, descritta inInizia a usare API Gateway, o chiamando ilRiferimenti API. Esistono diversi modi per chiamarloAPI. Includono l'utilizzo di AWS Command Line Interface (AWS CLI) o l'utilizzo di un AWS SDK. Inoltre, puoi abilitare API la creazione con AWS CloudFormation modelli o (nel caso di REST APIs e HTTPAPIs)APIEstensioni aperte per API Gateway.

Per un elenco delle regioni in cui API Gateway è disponibile, nonché gli endpoint del servizio di controllo associati, consulta Amazon API Gateway Endpoints and Quotas.

Chiamata di un gateway API API

Uno sviluppatore di app utilizza il componente del servizio API Gateway for API Execution`execute-api`, denominato, per richiamare un API componente creato o distribuito in API Gateway. Le entità di programmazione sottostanti vengono esposte dal creato. API Esistono diversi modi per chiamarloAPI. Per ulteriori informazioni, consultare Richiama le API REST in API Gateway e Richiama le API WebSocket .

Accesso a API Gateway

Puoi accedere ad Amazon API Gateway nei seguenti modi:

- AWS Management Console— AWS Management Console Fornisce un'interfaccia web per la creazione e la gestioneAPIs. Dopo aver completato i passaggi indicatiPrerequisiti, è possibile accedere alla console API Gateway all'indirizzohttps://console.aws.amazon.com/apigateway.
- AWS SDKs— Se utilizzi un linguaggio di programmazione che AWS fornisce un SDK for, puoi utilizzare un per accedere SDK a API Gateway. SDKssemplifica l'autenticazione, si integra facilmente con il tuo ambiente di sviluppo e fornisci l'accesso ai comandi API Gateway. Per ulteriori informazioni, consulta Strumenti per Amazon Web Services.
- APIGateway V1 e V2 APIs: se utilizzi un linguaggio di programmazione per cui non SDK è disponibile, consulta Amazon APIGateway Version 1 API Reference e Amazon Gateway Version 2 API Reference. API
- AWS Command Line Interface: per ulteriori informazioni, consulta Preparazione alla configurazione con AWS Command Line Interface nella Guida per l'utente di AWS Command Line Interface .
- AWS Tools for Windows PowerShell: per ulteriori informazioni, consulta Come configurare AWS Tools for Windows PowerShell nella Guida per l'utente di AWS Tools for Windows PowerShell .

Parte dell'infrastruttura serverless AWS

Oltre a AWS Lambda, API Gateway costituisce la parte dell'infrastruttura serverless rivolta alle AWS app. Per ulteriori informazioni su come iniziare a usare il serverless, consulta Serverless Developer Guide.

Affinché un'app richiami AWS servizi disponibili pubblicamente, puoi utilizzare Lambda per interagire con i servizi richiesti ed esporre le funzioni API Lambda tramite metodi in Gateway. API AWS Lambda esegue il codice su un'infrastruttura di elaborazione ad alta disponibilità. Esegue l'amministrazione e l'esecuzione necessarie delle risorse di calcolo. Per abilitare le applicazioni serverless, API Gateway supporta integrazioni proxy semplificate con AWS Lambda gli endpoint. HTTP

Come iniziare a usare Amazon API Gateway

Per un'introduzione ad Amazon API Gateway, consulta quanto segue:

- Inizia a usare, che fornisce una procedura dettagliata per creare un. HTTP API
- Terreno serverless, che fornisce video didattici.
- Happy Little API Shorts, che è una serie di brevi video didattici.

Concetti di Amazon API Gateway

La sezione seguente descrive i concetti introduttivi all'uso API di Gateway.

API Gateway

APIGateway è un AWS servizio che supporta quanto segue:

- Creazione, implementazione e gestione di un'interfaccia di programmazione delle RESTfulapplicazioni (API) per esporre HTTP endpoint, AWS Lambda funzioni o altri servizi di backend. AWS
- Creazione, implementazione e gestione di funzioni o altri servizi WebSocketAPIper AWS Lambda esporre. AWS
- Richiamo di API metodi esposti tramite il frontend e gli endpointHTTP. WebSocket

APIGateway REST API

Una raccolta di HTTP risorse e metodi integrati con HTTP endpoint di backend, funzioni Lambda o altri servizi. AWS È possibile distribuire questa raccolta in una o più fasi. In genere, API le risorse sono organizzate in un albero di risorse in base alla logica dell'applicazione. Ogni API risorsa può esporre uno o più API metodi con HTTP verbi univoci supportati da API Gateway. Per ulteriori informazioni, consulta the section called “Scegli tra REST APIs e HTTP APIs ”.

APIGateway HTTP API

Una raccolta di percorsi e metodi integrati con HTTP endpoint di backend o funzioni Lambda. È possibile distribuire questa raccolta in una o più fasi. Ogni route può esporre uno o più API metodi con HTTP verbi univoci supportati da Gateway. API Per ulteriori informazioni, consulta the section called “Scegli tra REST APIs e HTTP APIs ”.

APIGateway WebSocket API

Una raccolta di WebSocket percorsi e chiavi di percorso integrati con HTTP endpoint di backend, funzioni Lambda o altri servizi. AWS È possibile distribuire questa raccolta in una o più fasi. APIi metodi vengono richiamati tramite WebSocket connessioni frontend che è possibile associare a un nome di dominio personalizzato registrato.

APIdistribuzione

Un' point-in-time istantanea del tuo API GatewayAPI. Per essere disponibile per l'uso da parte dei client, l'implementazione deve essere associata a una o più API fasi.

Sviluppatore API

L' AWS account che possiede una distribuzione API Gateway (ad esempio, un fornitore di servizi che supporta anche l'accesso programmatico).

Endpoint API

Un nome host per un API gateway API interno distribuito in una regione specifica. Il formato del nome host è `{api-id}.execute-api.{region}.amazonaws.com`. Sono supportati i seguenti tipi di API endpoint:

- Endpoint ottimizzato per Edge API
- APIEndpoint privato
- Endpoint regionale API

Chiave API

Una stringa alfanumerica utilizzata da API Gateway per identificare uno sviluppatore di app che utilizza il tuo o. REST WebSocket API APIGateway può generare API chiavi per tuo conto oppure puoi importarle da un CSV file. Puoi utilizzare API le chiavi insieme agli autorizzatori Lambda o ai piani di utilizzo per controllare l'accesso al tuo. APIs

Vedi gli endpointAPI.

APIproprietario

Vedi APIsviluppatore.

APIpalco

Un riferimento logico allo stato del ciclo di vita del tuo API (ad esempio, 'dev', 'prod', 'beta', 'v2'). APIle fasi sono identificate dall'APIID e dal nome della fase.

Sviluppatore di app

Un creatore di app che può avere o meno un AWS account e che interagisce con API quello che tu, lo API sviluppatore, hai distribuito. Gli sviluppatori di app sono i tuoi clienti. Uno sviluppatore di app viene in genere identificato da una API chiave.

Richiamata URL

Quando si connette un nuovo client tramite una WebSocket connessione, è possibile chiamare un'integrazione in API Gateway per memorizzare il callback URL del client. È quindi possibile utilizzare tale callback URL per inviare messaggi al client dal sistema di backend.

Portale per sviluppatori

Un'applicazione che consente ai vostri clienti di registrarsi, scoprire e abbonarsi ai vostri API prodotti (piani di utilizzo API Gateway), gestire API le loro chiavi e visualizzare le metriche di utilizzo relative ai vostri prodotti. APIs

Endpoint ottimizzato per l'edge API

Il nome host predefinito di un API gateway API che viene distribuito nella regione specificata utilizzando una CloudFront distribuzione per facilitare l'accesso dei client, in genere da più regioni. AWS APIle richieste vengono indirizzate al CloudFront Point of Presence (POP) più vicino, il che in genere migliora i tempi di connessione per client geograficamente diversi.

Vedi gli endpoint. API

Richiesta di integrazione

L'interfaccia interna di una WebSocket API route o di un REST API metodo in API Gateway, in cui si mappa il corpo di una richiesta di route o i parametri e il corpo di una richiesta di metodo ai formati richiesti dal backend.

Risposta di integrazione

L'interfaccia interna di un WebSocket API percorso o di un REST API metodo in API Gateway, in cui si mappano i codici di stato, le intestazioni e il payload ricevuti dal backend al formato di risposta restituito a un'app client.

Modello di mappatura

Uno script in Velocity Template Language (VTL) che trasforma il corpo di una richiesta dal formato di dati di frontend al formato di dati di backend o che trasforma un corpo di risposta dal formato di dati di backend al formato di dati frontend. I modelli di mappatura possono essere specificati nella richiesta o nella risposta di integrazione. Possono fare riferimento ai dati disponibili al runtime come contesto e variabili di fase.

La mappatura può essere semplice come una trasformazione di identità che trasferisce le intestazioni o il corpo mediante l'integrazione senza alcuna modifica dal client al back-end per una richiesta. Lo stesso vale per una risposta, in cui il payload viene trasferito dal back-end al client.

Richiesta metodo

L'interfaccia pubblica di un API metodo in API Gateway che definisce i parametri e il corpo che uno sviluppatore di app deve inviare per le richieste di accesso al backend tramite. API

Risposta di metodo

L'interfaccia pubblica di a REST API che definisce i codici di stato, le intestazioni e i modelli di body che uno sviluppatore di app dovrebbe aspettarsi nelle risposte di. API

Integrazione fittizia

In un'integrazione fittizia, API le risposte vengono generate direttamente da API Gateway, senza la necessità di un backend di integrazione. In qualità di API sviluppatore, sei tu a decidere in che modo API Gateway risponde a una richiesta di integrazione fittizia. A tale scopo, è necessario configurare la richiesta e la risposta di integrazione del metodo per associare una risposta a un codice di stato specifico.

Modello

Schema di dati che specifica la struttura di dati di una richiesta o di un payload di risposta. È necessario un modello per generare un file fortemente tipizzatoSDK. API Viene anche utilizzato per convalidare payload. Un modello è utile per generare un modello di mappatura di esempio per avviare la creazione di un modello di mappatura di produzione. Sebbene sia utile, il modello non è richiesto per la creazione di un modello di mappatura.

Privato API

Vedi APIEndpoint privato.

Endpoint privato API

Un API endpoint esposto tramite VPC endpoint di interfaccia e che consente a un client di accedere in modo sicuro alle risorse private API all'interno di un. VPC APIsI privati sono isolati dalla rete Internet pubblica ed è possibile accedervi solo utilizzando gli VPC endpoint for API Gateway a cui è stato concesso l'accesso.

Integrazione privata

Un tipo di integrazione API Gateway che consente a un client di accedere alle risorse interne di un cliente VPC tramite un REST API endpoint privato senza esporre le risorse alla rete Internet pubblica.

Integrazione proxy

Una configurazione di integrazione API Gateway semplificata. Puoi configurare un'integrazione proxy come integrazione HTTP proxy o integrazione proxy Lambda.

Per l'integrazione con il HTTP proxy, API Gateway trasmette l'intera richiesta e risposta tra il frontend e un HTTP backend. Per l'integrazione del proxy Lambda, API Gateway invia l'intera

richiesta come input a una funzione Lambda di backend. APIGateway trasforma quindi l'output della funzione Lambda in una HTTP risposta frontend.

In RESTAPIs, l'integrazione proxy viene utilizzata più comunemente con una risorsa proxy, che è rappresentata da una variabile greedy path (ad esempio,`{proxy+}`) combinata con un metodo catch-allANY.

Creazione rapida

È possibile utilizzare la creazione rapida per semplificare la creazione di un. HTTP API La creazione rapida crea un percorso API con Lambda o HTTP integrazione, un percorso generico predefinito e uno stadio predefinito configurato per distribuire automaticamente le modifiche. Per ulteriori informazioni, consulta the section called “Crea un'API HTTP utilizzando la AWS CLI”.

Endpoint regionale API

Il nome host di un API dispositivo distribuito nella regione specificata e destinato a servire i client, ad esempio EC2 le istanze, nella stessa regione. AWS APIle richieste vengono indirizzate direttamente al API gateway specifico della regione API senza passare attraverso alcuna distribuzione. CloudFront Per le richieste all'interno della regione, un endpoint regionale aggira l'inutile processo di andata e ritorno verso una distribuzione. CloudFront

Inoltre, è possibile applicare il routing basato sulla latenza sugli endpoint regionali per distribuirlo su più regioni utilizzando la stessa configurazione degli API endpoint regionali, impostare lo stesso nome di dominio personalizzato per ogni dispositivo distribuito API e configurare i DNS record basati sulla latenza in Route 53 per indirizzare le richieste dei client verso la regione con la latenza più bassa. API

Vedi API gli endpoint.

Route

Una WebSocket route in API Gateway viene utilizzata per indirizzare i messaggi in arrivo a un'integrazione specifica, ad esempio una AWS Lambda funzione, in base al contenuto del messaggio. Quando definisci il tuo WebSocket API, specifichi una chiave di percorso e un backend di integrazione. La chiave route è un attributo nel corpo del messaggio. Quando viene trovata una corrispondenza per la chiave route in un messaggio in entrata, viene richiamato il back-end di integrazione.

È anche possibile impostare una route predefinita per chiavi route non corrispondenti o per specificare un modello di proxy che trasferisce il messaggio senza alcuna modifica ai componenti di back-end che eseguono l'instradamento ed elaborano la richiesta.

Richiesta di instradamento

L'interfaccia pubblica di un WebSocket API metodo in API Gateway che definisce il corpo che uno sviluppatore di app deve inviare nelle richieste di accesso al backend tramite. API

Risposta di instradamento

L'interfaccia pubblica di a WebSocket API che definisce i codici di stato, le intestazioni e i modelli di body che uno sviluppatore di app dovrebbe aspettarsi da API Gateway.

Piano di utilizzo

Un piano di utilizzo fornisce a API client selezionati l'accesso a uno o più o più dispositivi distribuitiREST. WebSocket APIs È possibile utilizzare un piano di utilizzo per configurare i limiti di limitazione e quota, che vengono applicati alle singole chiavi client. API

WebSocket connessione

APIGateway mantiene una connessione persistente tra i client e il API Gateway stesso. Non esiste una connessione persistente tra il API gateway e le integrazioni di backend come le funzioni Lambda. I servizi di back-end vengono richiamati in base alle esigenze, a seconda del contenuto dei messaggi ricevuti dai client.

Scegli tra REST APIs e HTTP APIs

RESTAPIse HTTP APIs sono entrambi RESTful API prodotti. RESTAPIssupportano un numero maggiore di funzionalità HTTPAPIs, mentre HTTP APIs sono progettati con funzionalità minime in modo da poter essere offerti a un prezzo inferiore. Scegli REST APIs se ti servono funzionalità come API chiavi, limitazione per client, convalida delle richieste, AWS WAF integrazione o endpoint privati. API Scegli HTTP APIs se non ti servono le funzionalità incluse in. REST APIs

Le seguenti sezioni riassumono le funzionalità principali disponibili in REST APIs e HTTPAPIs. Se necessario, vengono forniti collegamenti aggiuntivi per navigare tra le HTTP API sezioni REST API e della API Gateway Developer Guide.

Tipo di endpoint

Il tipo di endpoint si riferisce all'endpoint che API Gateway crea per te. API Per ulteriori informazioni, consulta the section called “Tipi di endpoint API Gateway”.

Tipi di endpoint	REST API	HTTP API
Ottimizzato per edge	Sì	No
Regionale	Sì	Sì
Privata	Sì	No

Sicurezza

APIGateway offre diversi modi per proteggervi API da determinate minacce, come malintenzionati o picchi di traffico. Per ulteriori informazioni, consultare the section called “Protezione” e the section called “Protezione”.

Funzionalità di sicurezza	REST API	HTTP API
Autenticazione reciproca TLS	Sì	Sì
Certificati per l'autenticazione di back-end	Sì	No
AWS WAF	Sì	No

Autorizzazione

APIGateway supporta diversi meccanismi per il controllo e la gestione dell'accesso al tuoAPI. Per ulteriori informazioni, consulta the section called “Controllo accessi” e the section called “Controllo accessi”.

Opzioni di autorizzazione	REST API	HTTP API
IAM	Sì	Sì
Policy delle risorse	Sì	No
Amazon Cognito	Sì	Sì [1]
Autorizzazione personalizzata con una AWS Lambda funzione	Sì	Sì
JSONToken Web (JWT [2])	No	Sì

[1] Puoi usare Amazon Cognito con un JWT autorizzatore.

[2] È possibile utilizzare un autorizzatore Lambda per la convalida. JWTs REST APIs

APIgestione

Scegli REST APIs se hai bisogno di funzionalità di API gestione come API chiavi e limitazione della tariffa per cliente. Per ulteriori informazioni, consulta the section called “Distribuzione”, the section called “Nomi di dominio personalizzati” e the section called “Nomi di dominio personalizzati”.

Funzionalità	REST API	HTTP API
Domini personalizzati	Sì	Sì
APIchiavi	Sì	No
Limitazione della frequenza per client	Sì	No

Funzionalità	REST API	HTTP API
Limitazione (della larghezza di banda della rete) dell'utilizzo per client	Sì	No

Sviluppo

Durante lo sviluppo del tuo API GatewayAPI, decidi in base a una serie di caratteristiche del tuoAPI. Queste caratteristiche dipendono dal caso d'uso del tuoAPI. Per ulteriori informazioni, consulta the section called “Sviluppare” e the section called “Sviluppare”.

Funzionalità	REST API	HTTP API
CORSconfigurazione	Sì	Sì
Chiamate di test	Sì	No
Caching	Sì	No
Implementazioni controllate dall'utente	Sì	Sì
Implementazioni automatiche	No	Sì
Risposte gateway personalizzate	Sì	No
Implementazione di una release Canary	Sì	No
Convalida delle richieste	Sì	No

Funzionalità	REST API	HTTP API
Trasformazione parametro della richiesta	Sì	Sì
Trasformazione corpo della richiesta	Sì	No

Monitoraggio

APIGateway supporta diverse opzioni per registrare API le richieste e monitorare le tueAPIs. Per ulteriori informazioni, consulta the section called “Monitoraggio” e the section called “Monitoraggio”.

Funzionalità	REST API	HTTP API
CloudWatch Metriche Amazon	Sì	Sì
Accedi ai log ai log CloudWatch	Sì	Sì
Log di accesso ad Amazon Data Firehose	Sì	No
Registri di esecuzione	Sì	No
AWS X-Ray tracciamento	Sì	No

Integrazioni

Le integrazioni collegano il tuo API Gateway API alle risorse di backend. Per ulteriori informazioni, consulta the section called “Integrazioni” e the section called “Integrazioni”.

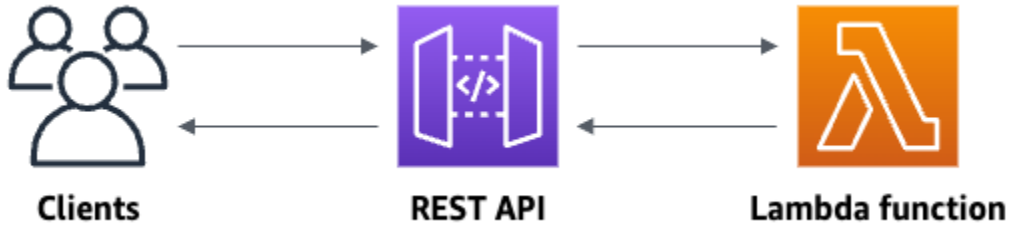
Funzionalità	REST API	HTTP API
Endpoint pubblici HTTP	Sì	Sì
AWS servizi	Sì	Sì
AWS Lambda funzioni	Sì	Sì
Integrazioni private con Network Load Balancer	Sì	Sì
Integrazioni private con il sistema di bilanciamento del carico dell'applicazione	No	Sì
Integrazioni private con AWS Cloud Map	No	Sì
Integrazioni fittizie	Sì	No

Inizia a usare la REST API console

In questo esercizio introduttivo, creerai un server serverless REST API utilizzando la console API Gateway RESTAPI. La tecnologia serverless APIs consente di concentrarsi sulle applicazioni anziché dedicare tempo al provisioning e alla gestione dei server. Il completamento di questa esercitazione richiede meno di 20 minuti e rientra nel piano gratuito AWS.

Innanzitutto, crei una funzione Lambda utilizzando la console Lambda. Successivamente, si crea una console REST API utilizzando la console API Gateway RESTAPI. Quindi, crei un API metodo e lo integri con una funzione Lambda utilizzando un'integrazione proxy Lambda. Infine, distribuisci e richiami il tuo. API

Quando richiami la tua RESTAPI, API Gateway indirizza la richiesta alla tua funzione Lambda. Lambda esegue la funzione e restituisce una risposta a API Gateway. APIGateway ti restituisce quindi la risposta.



Per completare questo esercizio, è necessario un utente Account AWS and an AWS Identity and Access Management (IAM) con accesso alla console. Per ulteriori informazioni, consulta Configurazione per l'utilizzo di API Gateway.

Argomenti

- Fase 1: creazione di una funzione Lambda
- Fase 2: Creare un REST API
- Fase 3: Creazione di un'integrazione proxy Lambda
- Fase 4: Implementa il API
- Passaggio 5: richiama il API
- (Opzionale) Fase 6: pulizia

Fase 1: creazione di una funzione Lambda

Usi una funzione Lambda per il backend del tuo. API Lambda esegue il codice solo quando è necessario e si dimensiona automaticamente, da poche richieste al giorno a migliaia al secondo.

Per questa esercitazione, utilizzi una funzione Node.js predefinita nella console Lambda.

Per creare una funzione Lambda

1. Accedi alla console Lambda all'indirizzo https://console.aws.amazon.com /lambda.
2. Scegli Crea funzione.
3. In Basic information (Informazioni di base) , per Function name (Nome funzione) , inserisci **`my-function`**.
4. Per tutte le altre opzioni, usa l'impostazione predefinita.

5. Scegli Crea funzione.

Il codice predefinito della funzione Lambda dovrebbe essere simile al seguente:

```
export const handler = async (event) => {
    const response = {
        statusCode: 200,
        body: JSON.stringify('The API Gateway REST API console is great!'),
    };
    return response;
};
```

È possibile modificare la funzione Lambda per questo esercizio, purché la risposta della funzione sia in linea con il formato richiesto da Gateway. API

Sostituisci il corpo della risposta predefinito (`Hello from Lambda!`) con `The API Gateway REST API console is great!`. Quando richiami la funzione di esempio, restituisce una risposta `200` ai client, insieme alla risposta aggiornata.

Fase 2: Creare un REST API

Successivamente, si crea un REST API file con una risorsa root (`/`).

Per creare un REST API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Esegui una di queste operazioni:
   - Per creare il tuo primo fileAPI, scegli Build. RESTAPI
   - Se ne hai già creato unoAPI, scegli Crea API, quindi scegli Crea per RESTAPI.
3. Per APInome, inserisci**`my-rest-api`**.
4. (Facoltativo) In Description (Descrizione), immettere una descrizione.
5. Mantieni il tipo di API endpoint impostato su Regionale.
6. Scegli Crea API.

Fase 3: Creazione di un'integrazione proxy Lambda

Successivamente, crei un API metodo per la tua REST API risorsa root (`/`) e integri il metodo con la tua funzione Lambda utilizzando un'integrazione proxy. In un'integrazione con proxy Lambda, API Gateway passa la richiesta in entrata dal client direttamente alla funzione Lambda.

Creazione di un'integrazione proxy Lambda

1. Seleziona la risorsa `/`, quindi scegli Crea metodo.
2. In Tipo di metodo, seleziona `ANY`.
3. In Tipo di integrazione, seleziona Lambda.
4. Attiva l'opzione Integrazione proxy Lambda.
5. In Funzione Lambda, immetti **`my-function`**, quindi seleziona la funzione Lambda.
6. Scegli Crea metodo.

Fase 4: Implementa il API

Successivamente, crei una API distribuzione e la associ a una fase.

Per distribuire il API

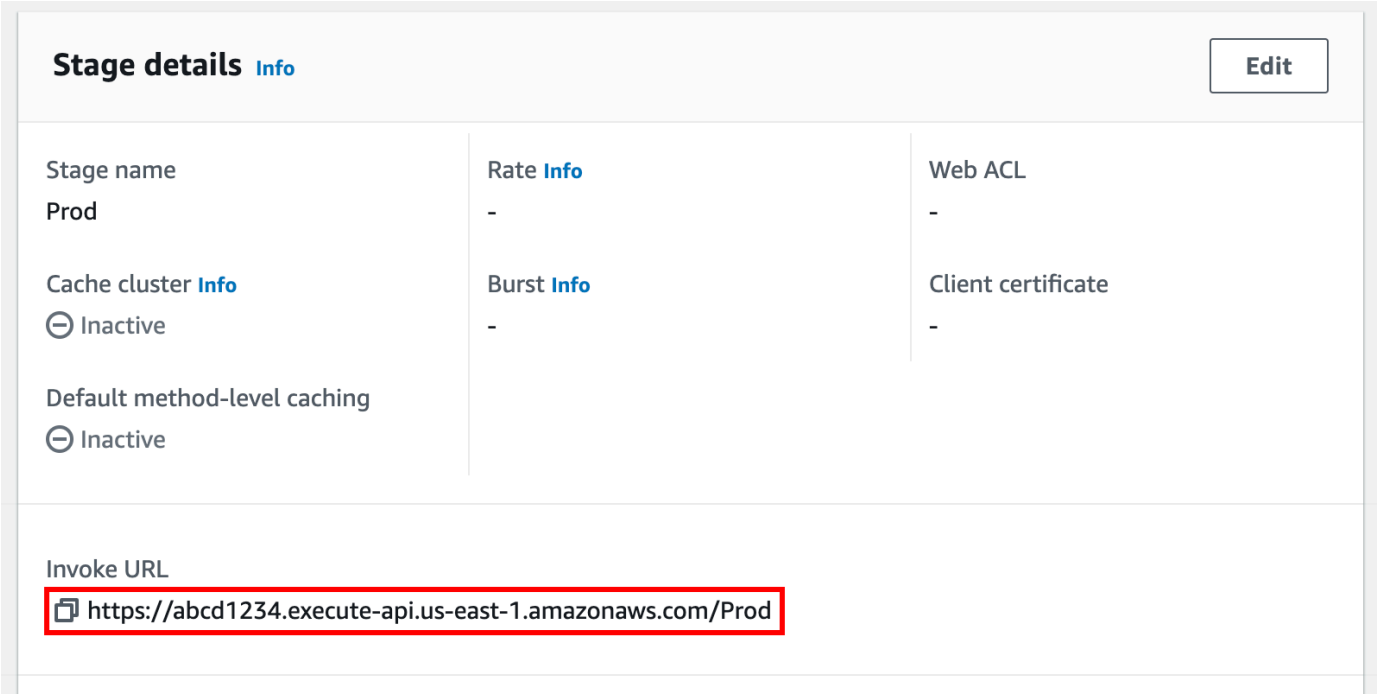
1. Scegliere Deploy (Implementa)API.
2. In Fase, seleziona Nuova fase.
3. In Stage name (Nome fase) immettere **`Prod`**.
4. (Facoltativo) In Description (Descrizione), immettere una descrizione.
5. Seleziona Deploy (Implementa).

Ora i clienti possono chiamare il tuoAPI. Per testarlo API prima di distribuirlo, puoi facoltativamente scegliere il ANYmetodo, accedere alla scheda Test e quindi scegliere Test.

Passaggio 5: richiama il API

Per invocare il tuo API

1. Nel riquadro di navigazione principale, seleziona Log.
2. In Dettagli dello stage, scegli l'icona di copia per copiare API la tua chiamataURL.



3. Inserisci l'invoke URL in un browser web.

   La versione completa URL dovrebbe `https://abcd123.execute-api.us-east-2.amazonaws.com/Prod` assomigliare a.

   Il tuo browser invia una `GET` richiesta aAPI.

4. Verifica la tua API risposta. Il testo `"The API Gateway REST API console is great!"` dovrebbe essere visualizzato nel browser in uso.

(Opzionale) Fase 6: pulizia

Per evitare di incorrere in costi inutili Account AWS, elimina le risorse che hai creato come parte di questo esercizio. I passaggi seguenti eliminano REST API la tua funzione Lambda e le risorse associate.

Per eliminare le tue REST API

1. Nel riquadro Risorse, scegli APIazioni, Elimina API.
2. Nella API finestra di dialogo Elimina, immettete conferma, quindi scegliete Elimina.

Eliminazione della funzione Lambda

1. Accedi alla console Lambda all'indirizzo https://console.aws.amazon.com /lambda.
2. Nella pagina Funzioni, seleziona la tua funzione. Scegli Operazioni > Elimina.
3. Nella finestra di dialogo Elimina 1 funzione/i, immetti **`delete`** e quindi scegli Elimina.

Eliminazione del gruppo di log della funzione Lambda

1. Apri la pagina dei gruppi di log della CloudWatch console Amazon.
2. Nella pagina Gruppi di log, seleziona il gruppo di log della funzione (`/aws/lambda/my-function`). In Operazioni, scegli Elimina gruppo di log.
3. Nella finestra di dialogo Delete log group(s) (Elimina gruppo/i di log) scegli Delete (Elimina).

Eliminazione del ruolo di esecuzione della funzione Lambda

1. Apri la pagina Ruoli della IAM console.
2. (Facoltativo) Nella pagina Ruoli, nella casella di ricerca, immetti **`my-function`**.
3. Seleziona il ruolo della tua funzione (ad esempio, `my-function-`*`31exxmpl`*), quindi scegli Elimina.
4. Nella finestra di dialogo Eliminare **`my-function-`***`31exxmpl`***?**, immetti il nome del ruolo, quindi scegli Elimina.

> **Tip**
>
> È possibile automatizzare la creazione e la pulizia delle AWS risorse utilizzando AWS CloudFormation o AWS Serverless Application Model ()AWS SAM. Per alcuni AWS CloudFormation modelli di esempio, consulta i modelli di esempio per API Gateway nel repository GitHub awsdocs.

Configurazione per l'utilizzo di API Gateway

Prima di utilizzare Amazon API Gateway per la prima volta, completa le seguenti attività.

Registrati per un Account AWS

Se non ne hai uno Account AWS, completa i seguenti passaggi per crearne uno.

Per iscriverti a un Account AWS

1. Apri la https://portal.aws.amazon.com/billing/registrazione.
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.

   Quando ti iscrivi a un Account AWS, Utente root dell'account AWSviene creato un. L'utente root dispone dell'accesso a tutte le risorse e tutti i Servizi AWS nell'account. Come best practice di sicurezza, assegna l'accesso amministrativo a un utente e utilizza solo l'utente root per eseguire attività che richiedono l'accesso di un utente root.

AWS ti invia un'email di conferma dopo il completamento della procedura di registrazione. In qualsiasi momento, puoi visualizzare l'attività corrente del tuo account e gestirlo accedendo a https://aws.amazon.com/e scegliendo Il mio account.

Crea un utente con accesso amministrativo

Dopo esserti registrato Account AWS, proteggi Utente root dell'account AWS AWS IAM Identity Center, abilita e crea un utente amministrativo in modo da non utilizzare l'utente root per le attività quotidiane.

Proteggi i tuoi Utente root dell'account AWS

1. Accedi AWS Management Consolecome proprietario dell'account scegliendo Utente root e inserendo il tuo indirizzo Account AWS email. Nella pagina successiva, inserisci la password.

   Per informazioni sull'accesso utilizzando un utente root, consulta la pagina Signing in as the root user della Guida per l'utente di Accedi ad AWS .

2. Attiva l'autenticazione a più fattori (MFA) per il tuo utente root.

   Per istruzioni, consulta Abilitare un MFA dispositivo virtuale per l'utente Account AWS root (console) nella Guida per l'IAMutente.

Crea un utente con accesso amministrativo

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 IAM Identity Center, concedi l'accesso amministrativo a un utente.

   Per un tutorial sull'utilizzo di IAM Identity Center directory come fonte di identità, consulta Configurare l'accesso utente con i valori predefiniti IAM Identity Center directory nella Guida per l'AWS IAM Identity Center utente.

Accesso come utente amministratore

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

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

Assegna l'accesso a ulteriori utenti

1. In IAM Identity Center, crea un set di autorizzazioni che segua la migliore pratica di applicazione delle autorizzazioni con privilegi minimi.

   Segui le istruzioni riportate nella pagina Creazione di un set di autorizzazioni nella Guida per l'utente di AWS IAM Identity Center .
2. Assegna al gruppo prima gli utenti e poi l'accesso con autenticazione unica (Single Sign-On).

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

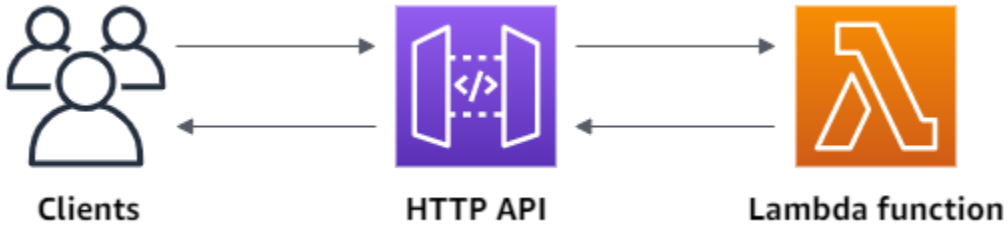
Inizia a usare API Gateway

In questo esercizio introduttivo, creerai un serverless. API La modalità serverless APIs consente di concentrarsi sulle applicazioni, anziché dedicare tempo al provisioning e alla gestione dei server. Il completamento di questo esercizio richiede meno di 20 minuti e può essere effettuato all'interno del piano gratuito AWS.

Innanzitutto, crei una funzione Lambda utilizzando la AWS Lambda console. Successivamente, ne crei una HTTP API utilizzando la console API Gateway. Quindi, invochi il tuoAPI.

> **Note**
>
> Questo esercizio utilizza un HTTPAPI. APIGateway supporta anche RESTAPIs, che includono più funzionalità. Per un tutorial sull'uso di a RESTAPI, vedithe section called “Inizia a usare la REST API console”.
> Per ulteriori informazioni sulla differenza tra HTTP APIs e RESTAPIs, vederethe section called “Scegli tra REST APIs e HTTP APIs ”.

Quando richiami la tua HTTPAPI, API Gateway indirizza la richiesta alla tua funzione Lambda. Lambda esegue la funzione Lambda e restituisce una risposta a Gateway. API APIGateway restituisce quindi una risposta all'utente.



Per completare questo esercizio, sono necessari un AWS account e un AWS Identity and Access Management utente con accesso alla console. Per ulteriori informazioni, consulta Prerequisiti.

Argomenti

- Fase 1: creazione di una funzione Lambda
- Fase 2: Creare un HTTP API
- Fase 3: Metti alla prova il tuo API

- (Opzionale) Fase 4: pulizia
- Passaggi successivi

Fase 1: creazione di una funzione Lambda

Usi una funzione Lambda per il backend del tuo. API Lambda esegue il codice solo quando è necessario e si dimensiona automaticamente, da poche richieste al giorno a migliaia al secondo.

Per questo esempio, viene utilizzata la funzione Node.js predefinita dalla console Lambda.

Per creare una funzione Lambda

1. Accedi alla console Lambda all'indirizzo https://console.aws.amazon.com /lambda.
2. Selezionare Create function (Crea funzione).
3. Nel campo Function name (Nome funzione), immettere **`my-function`**.
4. Per tutte le altre opzioni, usa l'impostazione predefinita.
5. Selezionare Create function (Crea funzione).

La funzione di esempio restituisce una risposta `200` ai client e il testo `Hello from Lambda!`.

Puoi modificare la tua funzione Lambda, purché la risposta della funzione sia in linea con il formato richiesto da Gateway. API

Il codice predefinito della funzione Lambda dovrebbe essere simile al seguente:

```
export const handler = async (event) => {
    const response = {
        statusCode: 200,
        body: JSON.stringify('Hello from Lambda!'),
    };
    return response;
};
```

Fase 2: Creare un HTTP API

Successivamente, crei un HTTPAPI. APIGateway supporta anche REST APIs e WebSocket APIs, ma an HTTP API è la scelta migliore per questo esercizio. RESTAPIssupporta più funzionalità

di HTTPAPIs, ma non abbiamo bisogno di tali funzionalità per questo esercizio. HTTPAPIssono progettati con funzionalità minime in modo da poter essere offerti a un prezzo inferiore. WebSocket APIsmantengono connessioni persistenti con i client per la comunicazione full-duplex, che non è richiesta in questo esempio.

HTTPAPIfornisce un HTTP endpoint per la funzione Lambda. APIIl gateway indirizza le richieste alla funzione Lambda e quindi restituisce la risposta della funzione ai client.

Per creare un HTTP API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Esegui una di queste operazioni:
   - Per creare il tuo primo fileAPI, scegli Build. HTTPAPI
   - Se ne hai già creato unoAPI, scegli Crea API, quindi scegli Crea per HTTPAPI.
3. Per Integrazioni, scegliere Aggiungi integrazione.
4. Selezionare Lambda.
5. Per Lambda function (Funzione Lambda), immetti **`my-function`**.
6. Per APInome, inserisci**`my-http-api`**.
7. Scegli Next (Successivo).
8. Controlla il percorso creato da API Gateway per te, quindi scegli Avanti.
9. Controlla la fase che API Gateway crea per te, quindi scegli Avanti.
10. Scegli Create (Crea) .

Ora ne hai creato uno HTTP API con un'integrazione Lambda pronto a ricevere richieste dai clienti.

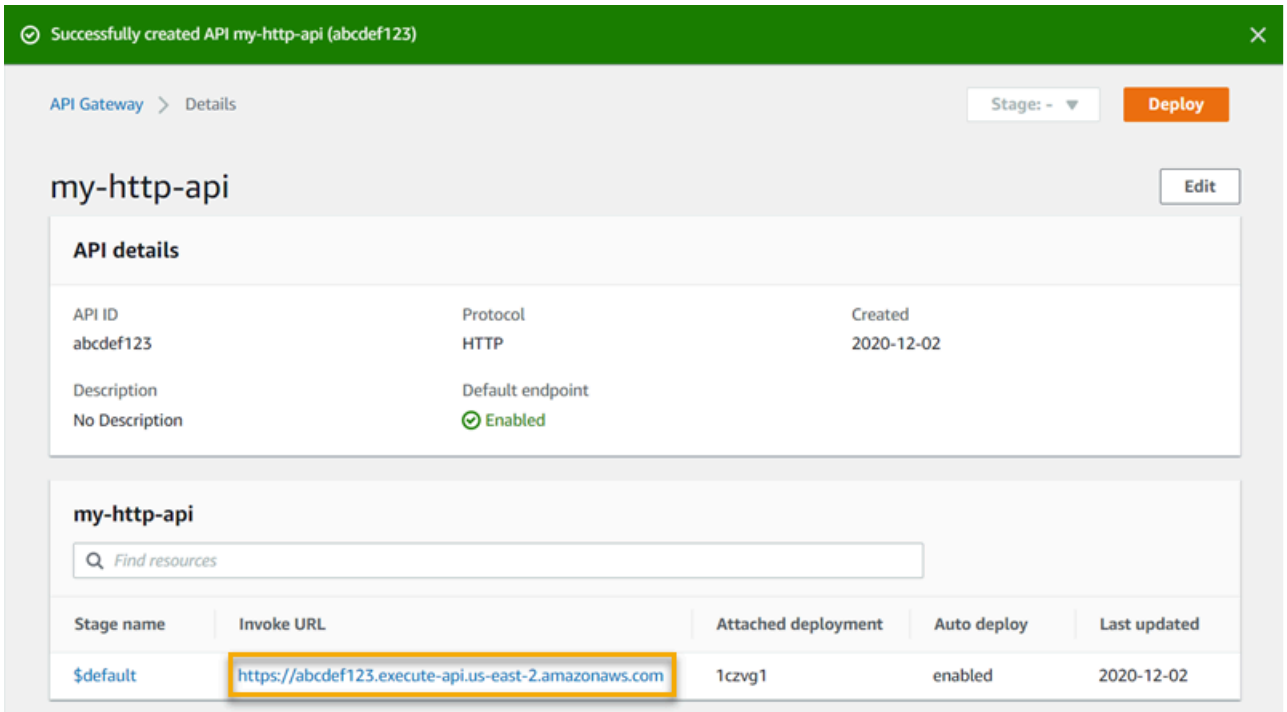
Fase 3: Metti alla prova il tuo API

Successivamente, testate il vostro API per assicurarvi che funzioni. Per semplicità, usa un browser web per richiamare il tuoAPI.

Per testare il tuo API

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. APIScegli il tuo.

3. Annota la tua API URL invocazione.



4. Copia API la tua chiamata e URL inseriscila in un browser web. Aggiungi il nome della tua funzione Lambda all'URLinvoke per chiamare la tua funzione Lambda. Per impostazione predefinita, la console API Gateway crea una route con lo stesso nome della funzione Lambda,. `my-function`

   La versione completa URL dovrebbe `https://`*`abcdef123`*`.execute-api.`*`us-east-2`*`.amazonaws.com/`*`my-function` assomigliare a.

   Il tuo browser invia una `GET` richiesta aAPI.

5. Verifica la tua API risposta. Il testo `"Hello from Lambda!"` dovrebbe essere visualizzato nel browser in uso.

(Opzionale) Fase 4: pulizia

Per evitare costi non necessari, eliminare le risorse create nell'ambito di questo esercizio di nozioni di base. I passaggi seguenti eliminano HTTP API la tua funzione Lambda e le risorse associate.

Per eliminare un HTTP API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nella APIspagina, seleziona un. API Scegli Azioni, quindi Elimina.

3. Scegliere Delete (Elimina).

Per eliminare una funzione Lambda

1. Accedi alla console Lambda all'indirizzo https://console.aws.amazon.com /lambda.
2. Nella pagina Funzioni, selezionare una funzione. Scegli Azioni, quindi Elimina.
3. Scegliere Delete (Elimina).

Per eliminare il gruppo di log di una funzione Lambda

1. Nella CloudWatch console Amazon, apri la pagina dei gruppi di log.
2. Nella pagina Log Groups (Gruppi di log), seleziona il gruppo di log della funzione (`/aws/lambda/my-function`). Scegliere Actions (Operazioni), quindi selezionare Delete log group (Elimina gruppo di log).
3. Scegliere Delete (Elimina).

Per eliminare il ruolo di esecuzione di una funzione Lambda

1. Nella AWS Identity and Access Management console, apri la pagina Ruoli.
2. Seleziona il ruolo della funzione, ad esempi, `my-function-`*`31exxmpl`*.
3. Scegliere Delete role (Elimina ruolo).
4. Scegliere Yes, delete (Sì, elimina).

È possibile automatizzare la creazione e la pulizia delle AWS risorse utilizzando AWS CloudFormation o. AWS SAM Per un esempio dei modelli AWS CloudFormation , consulta i modelli di esempio AWS CloudFormation.

Passaggi successivi

In questo esempio, hai usato il per AWS Management Console creare un file semplice. HTTP API HTTPAPIrichiama una funzione Lambda e restituisce una risposta ai client.

Di seguito sono riportati i passaggi successivi man mano che si continua a lavorare con Gateway. API

- Configura altri tipi di API integrazioni, tra cui:

  - HTTPendpoint
  - Risorse private in aVPC, come ECS i servizi Amazon
  - AWS servizi come Amazon Simple Queue Service e Kinesis Data Streams AWS Step Functions
- Controlla l'accesso al tuo APIs
- Abilita la registrazione per il tuo APIs
- Configura il throttling per il tuo APIs
- Configura domini personalizzati per i tuoi APIs

Per ricevere assistenza con Amazon API Gateway dalla community, consulta il APIGateway Discussion Forum. Quando entri in questo forum, AWS potrebbe essere necessario effettuare l'accesso.

Per ricevere assistenza con API Gateway direttamente da AWS, consulta le opzioni di AWS supporto nella pagina Supporto.

Consulta anche le nostre domande frequenti (FAQs) o contattaci direttamente.

Tutorial e workshop di Amazon API Gateway

I seguenti tutorial e workshop forniscono esercizi pratici per aiutarti a usare API Gateway.

Tutorial sull'API REST

- Scegli un tutorial di integrazione AWS Lambda
- Tutorial: creazione di un'API REST mediante l'importazione di un esempio
- Scegli un tutorial sull'integrazione HTTP
- Tutorial: crea un'API REST con un'integrazione privata
- Tutorial: creare un'API REST con un' AWS integrazione
- Tutorial: crea una calcolatrice REST API con due integrazioni di AWS servizi e un'integrazione non proxy Lambda
- Tutorial: creare un REST API proxy come Amazon S3
- Tutorial: creazione di un'API REST come un proxy Amazon Kinesis
- Tutorial: crea un'API REST utilizzando AWS SDK o AWS CLI
- Tutorial: crea un account privato REST API

Tutorial sull'API HTTP

- Tutorial: Creare un'API HTTP CRUD con Lambda e DynamoDB
- Tutorial: crea un HTTP API account con un'integrazione privata con un ECS servizio Amazon

WebSocket Tutorial sulle API

- Tutorial: crea un'app di WebSocket chat con a WebSocket API, Lambda e DynamoDB

Workshop

- Creazione di un'applicazione Web serverless
- CI/CD per applicazioni serverless
- Workshop sulla sicurezza serverless
- Gestione, autenticazione e autorizzazione delle identità serverless

- Workshop su Gateway Amazon API

RESTAPITutorial Amazon API Gateway

I seguenti tutorial forniscono esercizi pratici per aiutarti a conoscere Gateway. API REST APIs

Argomenti

- Scegli un tutorial di integrazione AWS Lambda
- Tutorial: creazione di un'API REST mediante l'importazione di un esempio
- Scegli un tutorial sull'integrazione HTTP
- Tutorial: crea un'API REST con un'integrazione privata
- Tutorial: creare un'API REST con un' AWS integrazione
- Tutorial: crea una calcolatrice REST API con due integrazioni di AWS servizi e un'integrazione non proxy Lambda
- Tutorial: creare un REST API proxy come Amazon S3
- Tutorial: creazione di un'API REST come un proxy Amazon Kinesis
- Tutorial: crea un'API REST utilizzando AWS SDK o AWS CLI
- Tutorial: crea un account privato REST API

Scegli un tutorial di integrazione AWS Lambda

Per creare un'integrazione API con Lambda, puoi utilizzare l'integrazione proxy Lambda o l'integrazione non proxy Lambda.

Nell'integrazione del proxy Lambda, l'input della funzione Lambda può essere espresso come qualsiasi combinazione di intestazioni di richiesta, variabili di percorso, parametri della stringa di query, corpo e dati di configurazione. API Devi solo scegliere una funzione Lambda. APIGateway configura automaticamente la richiesta di integrazione e la risposta di integrazione. Una volta configurato, il API metodo può evolversi senza modificare le impostazioni esistenti. Ciò è possibile perché la funzione Lambda di backend analizza i dati della richiesta in entrata e risponde al client.

Con l'integrazione non proxy Lambda, devi accertarti che l'input della funzione Lambda venga fornito come payload della richiesta di integrazione. È necessario mappare tutti i dati di input forniti dal client come parametri di richiesta nel corpo della richiesta di integrazione appropriato. Potrebbe anche

essere necessario convertire il corpo della richiesta fornito dal client in un formato riconosciuto dalla funzione Lambda.

In un proxy Lambda o in un'integrazione non proxy Lambda, puoi utilizzare una funzione Lambda in un account diverso da quello in cui hai creato il tuo. API

Argomenti

- Tutorial: Creare un'integrazione REST API con un proxy Lambda
- Tutorial: Creare un'integrazione non proxy REST API con Lambda
- Tutorial: Creazione di un'RESTAPIintegrazione proxy Lambda con più account

Tutorial: Creare un'integrazione REST API con un proxy Lambda

L'integrazione proxy Lambda è un tipo di API integrazione API Gateway leggero e flessibile che consente di integrare un API metodo, o un intero, con una API funzione Lambda. La funzione Lambda può essere scritta in qualsiasi linguaggio supportato da Lambda. Poiché si tratta di un'integrazione proxy, puoi modificare l'implementazione della funzione Lambda in qualsiasi momento senza dover ridistribuire la tua. API

In questo tutorial, esegui quanto indicato di seguito:

- Creazione di una funzione Lambda "Hello, World!" La funzione Lambda fungerà da backend per. API
- Creazione e test di un'API "Hello World!" APIcon l'integrazione del proxy Lambda.

Argomenti

- Creazione di una funzione Lambda "Hello, World!" valida e completa
- Crea un messaggio «Hello, World!» API
- Implementa e testa il API

Creazione di una funzione Lambda "Hello, World!" valida e completa

Per creare una funzione Lambda "Hello, World!" Funzione Lambda nella console Lambda

1. Accedi alla console Lambda all'indirizzo https://console.aws.amazon.com /lambda.
2. Nella barra di AWS navigazione, scegli un. Regione AWS

> **Note**
>
> Prendere nota della regione in cui si crea la funzione Lambda. Ti servirà quando creerai ilAPI.

3. Nel riquadro di navigazione scegliere Functions (Funzioni).
4. Selezionare Create function (Crea funzione).
5. Scegli Author from scratch (Crea da zero).
6. In Basic information (Informazioni di base) eseguire queste operazioni:
   a. In Function name (Nome funzione) immettere **`GetStartedLambdaProxyIntegration`**.
   b. Per Runtime, scegli l'ultimo runtime supportato di Node.js o di Python.
   c. Per Architecture, mantieni l'impostazione predefinita.
   d. In Autorizzazioni espandere Modifica ruolo di esecuzione predefinito. Nell'elenco a discesa Ruolo di esecuzione, scegli Crea nuovo ruolo dai modelli di policy AWS .
   e. In Role name (Nome ruolo) immettere **`GetStartedLambdaBasicExecutionRole`**.
   f. Lasciare il campo Policy templates (Modelli di policy) vuoto.
   g. Selezionare Create function (Crea funzione).
7. In Function code (Codice funzione), nell'editor di codice inline, copiare/incollare il codice seguente:

   Node.js

```
export const handler = function(event, context, callback) {
    console.log('Received event:', JSON.stringify(event, null, 2));
    var res ={
        "statusCode": 200,
        "headers": {
            "Content-Type": "*/*"
        }
    };
    var greeter = 'World';
    if (event.greeter && event.greeter!==="") {
        greeter =  event.greeter;
    } else if (event.body && event.body !== "") {
        var body = JSON.parse(event.body);
        if (body.greeter && body.greeter !== "") {
```

```
            greeter = body.greeter;
        }
    } else if (event.queryStringParameters &&
 event.queryStringParameters.greeter && event.queryStringParameters.greeter !==
 "") {
        greeter = event.queryStringParameters.greeter;
    } else if (event.multiValueHeaders && event.multiValueHeaders.greeter &&
 event.multiValueHeaders.greeter != "") {
        greeter = event.multiValueHeaders.greeter.join(" and ");
    } else if (event.headers && event.headers.greeter && event.headers.greeter !
= "") {
        greeter = event.headers.greeter;
    }

    res.body = "Hello, " + greeter + "!";
    callback(null, res);
};
```

Python

```
import json


def lambda_handler(event, context):
    print(event)

    greeter = 'World'

    try:
        if (event['queryStringParameters']) and (event['queryStringParameters']
['greeter']) and (
                event['queryStringParameters']['greeter'] is not None):
            greeter = event['queryStringParameters']['greeter']
    except KeyError:
        print('No greeter')

    try:
        if (event['multiValueHeaders']) and (event['multiValueHeaders']
['greeter']) and (
                event['multiValueHeaders']['greeter'] is not None):
            greeter = " and ".join(event['multiValueHeaders']['greeter'])
    except KeyError:
        print('No greeter')
```

```
    try:
        if (event['headers']) and (event['headers']['greeter']) and (
                event['headers']['greeter'] is not None):
            greeter = event['headers']['greeter']
    except KeyError:
        print('No greeter')

    if (event['body']) and (event['body'] is not None):
        body = json.loads(event['body'])
        try:
            if (body['greeter']) and (body['greeter'] is not None):
                greeter = body['greeter']
        except KeyError:
            print('No greeter')

    res = {
        "statusCode": 200,
        "headers": {
            "Content-Type": "*/*"
        },
        "body": "Hello, " + greeter + "!"
    }

    return res
```

8. Selezionare Deploy (Distribuisci).

Crea un messaggio «Hello, World!» API

Ora creane uno API per il tuo «Hello, World!» Funzione Lambda utilizzando la console API Gateway.

Per creare un messaggio «Hello, World!» API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Se è la prima volta che utilizzi API Gateway, viene visualizzata una pagina che presenta le funzionalità del servizio. In RESTAPI, scegli Costruisci. Quando viene visualizzato il API popup Crea esempio, scegli OK.

   Se non è la prima volta che usi API Gateway, scegli Crea API. In RESTAPI, scegli Build.
3. Per APInome, inserisci**`LambdaProxyAPI`**.
4. (Facoltativo) In Description (Descrizione), immettere una descrizione.

5. Mantieni il tipo di API endpoint impostato su Regionale.
6. Scegli Crea API.

Dopo aver creato unAPI, crei una risorsa. In genere, API le risorse sono organizzate in un albero di risorse in base alla logica dell'applicazione. In questo esempio crei una risorsa /helloworld.

Per creare una risorsa

1. Scegli Crea risorsa.
2. Mantieni l'opzione Risorsa proxy disattivata.
3. Mantieni Percorso risorsa impostato su `/`.
4. Per Resource Name (Nome risorsa) immetti **`helloworld`**.
5. Mantieni CORS(Cross Origin Resource Sharing) disattivato.
6. Scegli Crea risorsa.

In un'integrazione proxy, l'intera richiesta viene inviata alla funzione Lambda di backend così com'è, tramite un metodo generico che rappresenta qualsiasi `ANY` metodo. HTTP Il HTTP metodo effettivo viene specificato dal client in fase di esecuzione. Il `ANY` metodo consente di utilizzare un'unica configurazione di API metodo per tutti i HTTP metodi supportati: `DELETEGET`,`HEAD`,`OPTIONS`,`PATCH`,`POST`, e`PUT`.

Per creare un metodo **`ANY`**

1. Seleziona la risorsa /helloworld, quindi scegli Crea metodo.
2. In Tipo di metodo, seleziona ANY.
3. Per Tipo di integrazione seleziona Funzione Lambda.
4. Attiva l'opzione Integrazione proxy Lambda.
5. Per la funzione Lambda, seleziona il Regione AWS luogo in cui hai creato la funzione Lambda, quindi inserisci il nome della funzione.
6. Per utilizzare il valore di timeout predefinito di 29 secondi, mantieni attiva l'opzione Timeout predefinito. Per impostare un timeout personalizzato, scegli Timeout predefinito e immetti un valore di timeout compreso tra `50` e `29000` millisecondi.
7. Scegli Crea metodo.

Implementa e testa il API

Per implementare il tuo API

1. Scegliere Deploy (Implementa)API.
2. In Fase, seleziona Nuova fase.
3. In Stage name (Nome fase) immettere **`test`**.
4. (Facoltativo) In Description (Descrizione), immettere una descrizione.
5. Seleziona Deploy (Implementa).
6. In Stage details, scegli l'icona di copia per copiare API la tua chiamataURL.

Usa browser e c URL per testare an API con l'integrazione del proxy Lambda

Puoi usare un browser o c URL per testare il tuoAPI.

Per testare `GET` le richieste utilizzando solo i parametri della stringa di query, puoi inserire la `helloworld` risorsa URL for the API nella barra degli indirizzi del browser.

Per creare la risorsa URL for API the, aggiungi la `helloworld` risorsa `helloworld` e il parametro della stringa di query `?greeter=John` alla tua URL invoke. Il tuo URL dovrebbe avere il seguente aspetto.

```
https://r275xc9bmd.execute-api.us-east-1.amazonaws.com/test/helloworld?greeter=John
```

Per altri metodi, è necessario utilizzare utilità di REST API test più avanzate, come POSTMANo c URL. Questo tutorial utilizza c. URL Gli esempi di URL comandi c riportati di seguito presuppongono che c URL sia installato sul computer.

Per testare il tuo distribuito API usando c: URL

1. Apri una finestra del terminale.
2. Copia il seguente URL comando c e incollalo nella finestra del terminale, quindi sostituisci l'invoke URL con quello che hai copiato nel passaggio precedente e aggiungi **`/helloworld`** alla fine di. URL

> **Note**
>
> Se si esegue il comando in Windows, utilizzare questa sintassi:

```
curl -v -X POST "https://r275xc9bmd.execute-api.us-east-1.amazonaws.com/
test/helloworld" -H "content-type: application/json" -d "{ \"greeter\":
 \"John\" }"
```

a. Per chiamare il parametro API con la stringa di interrogazione di: `?greeter=John`

```
curl -X GET 'https://r275xc9bmd.execute-api.us-east-1.amazonaws.com/test/
helloworld?greeter=John'
```

b. Per chiamarlo API con un parametro header pari a: `greeter:John`

```
curl -X GET https://r275xc9bmd.execute-api.us-east-1.amazonaws.com/test/
helloworld \
  -H 'content-type: application/json' \
  -H 'greeter: John'
```

c. Per chiamarli API con un corpo di`{"greeter":"John"}`:

```
curl -X POST https://r275xc9bmd.execute-api.us-east-1.amazonaws.com/test/
helloworld \
  -H 'content-type: application/json' \
  -d '{ "greeter": "John" }'
```

In tutti i casi, l'output è una risposta 200 con il corpo seguente:

```
Hello, John!
```

Tutorial: Creare un'integrazione non proxy REST API con Lambda

In questa procedura dettagliata, utilizziamo la console API Gateway per creare una console API che consenta a un client di richiamare le funzioni Lambda tramite l'integrazione non proxy Lambda (nota anche come integrazione personalizzata). Per ulteriori informazioni su AWS Lambda e sulle funzioni Lambda, consulta la Guida per gli sviluppatori di AWS Lambda.

Per facilitare l'apprendimento, abbiamo scelto una semplice funzione Lambda con una API configurazione minima per guidarti attraverso i passaggi della creazione di un API gateway API con

l'integrazione personalizzata Lambda. Laddove necessario, descriveremo la logica. Per un esempio più dettagliato dell'integrazione personalizzata Lambda, consulta Tutorial: crea una calcolatrice REST API con due integrazioni di AWS servizi e un'integrazione non proxy Lambda.

Prima di creare ilAPI, configura il backend Lambda creando una funzione Lambda in, descritta di seguito. AWS Lambda

Argomenti

- Creazione di una funzione Lambda per l'integrazione non proxy Lambda
- Crea un'integrazione API senza proxy con Lambda
- Prova a richiamare il metodo API
- Implementa il API
- Testatela API in una fase di distribuzione
- Elimini

Creazione di una funzione Lambda per l'integrazione non proxy Lambda

> **Note**
>
> La creazione di funzioni Lambda può comportare addebiti sul tuo AWS account.

In questa fase, viene creata una funzione Lambda "Hello, World!" per l'integrazione Lambda personalizzata. In tutta la procedura guidata la funzione è denominata `GetStartedLambdaIntegration`.

L'implementazione di questa funzione Lambda `GetStartedLambdaIntegration` è la seguente:

Node.js

```
'use strict';
var days = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
 'Saturday'];
var times = ['morning', 'afternoon', 'evening', 'night', 'day'];

console.log('Loading function');

export const handler = function(event, context, callback) {
```

```
  // Parse the input for the name, city, time and day property values
  let name = event.name === undefined ? 'you' : event.name;
  let city = event.city === undefined ? 'World' : event.city;
  let time = times.indexOf(event.time)<0 ? 'day' : event.time;
  let day = days.indexOf(event.day)<0 ? null : event.day;

  // Generate a greeting
  let greeting = 'Good ' + time + ', ' + name + ' of ' + city + '. ';
  if (day) greeting += 'Happy ' + day + '!';

  // Log the greeting to CloudWatch
  console.log('Hello: ', greeting);

  // Return a greeting to the caller
  callback(null, {
      "greeting": greeting
  });
};
```

Python

```
import json

days = {
    'Sunday',
    'Monday',
    'Tuesday',
    'Wednesday',
    'Thursday',
    'Friday',
    'Saturday'}
times = {'morning', 'afternoon', 'evening', 'night', 'day'}


def lambda_handler(event, context):
    print(event)
    # parse the input for the name, city, time, and day property values
    try:
        if event['name']:
            name = event['name']
    except KeyError:
        name = 'you'
    try:
```

```
        if event['city']:
            city = event['city']
    except KeyError:
        city = 'World'
    try:
        if event['time'] in times:
            time = event['time']
        else:
            time = 'day'
    except KeyError:
        time = 'day'
    try:
        if event['day'] in days:
            day = event['day']
        else:
            day = ''
    except KeyError:
        day = ''
    # Generate a greeting
    greeting = 'Good ' + time + ', ' + name + ' of ' + \
        city + '.' + ['', ' Happy ' + day + '!'][day != '']
    # Log the greeting to CloudWatch
    print(greeting)

    # Return a greeting to the caller
    return {"greeting": greeting}
```

Per l'integrazione personalizzata Lambda, API Gateway passa l'input alla funzione Lambda dal client come corpo della richiesta di integrazione. L'oggetto `event` del gestore della funzione Lambda è l'input.

La nostra funzione Lambda è semplice. Analizza l'oggetto `event` di input per le proprietà `name`, `city`, `time` e `day`. Quindi restituisce un messaggio di saluto, come JSON oggetto di`{"message":greeting}`, al chiamante. Il messaggio è nel modello `"Good [morning|afternoon|day], [name|you] in [city|World]. Happy day!"`. Si presume che l'input della funzione Lambda provenga dal seguente oggetto: JSON

```
{
  "city": "...",
  "time": "...",
  "day": "...",
```

```
  "name" : "..."
}
```

Per ulteriori informazioni, consulta la Guida per gli sviluppatori di AWS Lambda.

Inoltre, la funzione registra la sua esecuzione su Amazon CloudWatch `console.log(...)` chiamando. Questo risulta utile per tracciare le chiamate durante il debug della funzione. Per consentire alla `GetStartedLambdaIntegration` funzione di registrare la chiamata, imposta un IAM ruolo con politiche appropriate per la funzione Lambda per creare gli CloudWatch stream e aggiungere voci di registro agli stream. La console Lambda ti guida nella creazione dei IAM ruoli e delle policy richiesti.

Se si configura API senza utilizzare la console API Gateway, ad esempio quando si importa un file API da un API file Open, è necessario creare e impostare in modo esplicito, se necessario, un ruolo e una politica di chiamata per API Gateway per richiamare le funzioni Lambda. Per ulteriori informazioni su come configurare i ruoli di chiamata ed esecuzione Lambda per un API gateway, vedere. API Controlla l'accesso a un REST API con IAM autorizzazioni

Rispetto alla `GetStartedLambdaProxyIntegration` funzione Lambda per l'integrazione del proxy Lambda, la funzione Lambda per l'integrazione personalizzata `GetStartedLambdaIntegration` Lambda riceve solo input dal corpo della richiesta di integrazione del Gateway. API API La funzione può restituire l'output di qualsiasi JSON oggetto, una stringa, un numero, un blob booleano o anche un blob binario. La funzione Lambda per l'integrazione del proxy Lambda, al contrario, può ricevere l'input da qualsiasi dato di richiesta, ma deve restituire l'output di un particolare oggetto. JSON La `GetStartedLambdaIntegration` funzione per l'integrazione personalizzata Lambda può avere i parametri di API richiesta come input, a condizione che API Gateway mappi i parametri di API richiesta richiesti al corpo della richiesta di integrazione prima di inoltrare la richiesta del client al backend. A tal fine, lo API sviluppatore deve creare un modello di mappatura e configurarlo sul metodo utilizzato durante la API creazione di. API

Viene ora creata la funzione Lambda `GetStartedLambdaIntegration`.

Per creare la funzione Lambda **`GetStartedLambdaIntegration`** per l'integrazione personalizzata Lambda

1. Apri la AWS Lambda console all'indirizzo https://console.aws.amazon.com/lambda/.
2. Esegui una di queste operazioni:

- Se viene visualizzata la pagina di benvenuto, scegliere Get Started Now (Inizia subito), quindi Create a function (Crea una funzione).
- Se viene visualizzata la pagina di elenco Lambda > Funzioni, scegliere Create a function (Crea una funzione).

3. Scegli Author from scratch (Crea da zero).
4. Nel riquadro Author from scratch (Crea da zero) procedere nel seguente modo:
   a. In Nome (Name), immettere **`GetStartedLambdaIntegration`** come nome della funzione Lambda.
   b. Per Runtime, scegli l'ultimo runtime supportato di Node.js o di Python.
   c. Per Architecture, mantieni l'impostazione predefinita.
   d. In Autorizzazioni espandere Modifica ruolo di esecuzione predefinito. Nell'elenco a discesa Ruolo di esecuzione, scegli Crea nuovo ruolo dai modelli di policy AWS .
   e. Per Role name (Nome ruolo), digitare un nome per il ruolo, ad esempio **`GetStartedLambdaIntegrationRole`**.
   f. Per Modelli di policy, scegliere Autorizzazioni microservizi semplici.
   g. Selezionare Create function (Crea funzione).
5. Nel riquadro Configure function (Configura funzione) effettua quanto segue in Function code (Codice funzione):
   a. Copiare il codice di funzione Lambda elencato all'inizio di questa sezione e incollarlo nell'editor di codice inline.
   b. Lascia i valori predefiniti negli altri campi di questa sezione.
   c. Selezionare Deploy (Distribuisci).
6. Per testare la funzione appena creata, scegli la scheda Test.
   a. Per Event name (Nome evento) immettere **`HelloWorldTest`**.
   b. Per Event JSON, sostituisci il codice predefinito con il seguente.

```
{
  "name": "Jonny",
  "city": "Seattle",
  "time": "morning",
  "day": "Wednesday"
}
```

c. Scegliere Test (Testa) per invocare la funzione. Viene visualizzata la sezione Execution result: succeeded (Risultato esecuzione: riuscito). Espandi Dettaglio. Verrà visualizzato l'output seguente.

```
{
    "greeting": "Good morning, Jonny of Seattle. Happy Wednesday!"
}
```

L'output viene anche scritto in CloudWatch Logs.

Come esercizio collaterale, puoi usare la IAM console per visualizzare il IAM ruolo (`GetStartedLambdaIntegrationRole`) creato come parte della creazione della funzione Lambda. A questo IAM ruolo sono associate due politiche in linea. Una stipula le autorizzazioni di base per l'esecuzione Lambda Consente di richiamare qualsiasi CloudWatch risorsa del tuo account nella regione in cui viene creata la funzione Lambda. CloudWatch `CreateLogGroup` Questa politica consente inoltre di creare CloudWatch flussi e registrare gli eventi per la funzione Lambda`GetStartedLambdaIntegration`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "logs:CreateLogGroup",
            "Resource": "arn:aws:logs:region:account-id:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:log-group:/aws/lambda/
GetStartedLambdaIntegration:*"
            ]
        }
    ]
}
```

L'altro documento di policy si applica all'invocazione di un altro AWS servizio che non viene utilizzato in questo esempio. Puoi ignorarlo in questa fase.

Al IAM ruolo è associata un'entità attendibile, che è`lambda.amazonaws.com`. La relazione di fiducia è la seguente:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

La combinazione di questa relazione di fiducia e della politica in linea consente alla funzione Lambda di richiamare `console.log()` una funzione per registrare gli eventi nei registri. CloudWatch

Crea un'integrazione API senza proxy con Lambda

Con la funzione Lambda (`GetStartedLambdaIntegration`) creata e testata, sei pronto per esporre la funzione tramite un gateway. API API A scopo illustrativo, esponiamo la funzione Lambda con un metodo generico. HTTP Utilizziamo il corpo della richiesta, una variabile di URL percorso, una stringa di query e un'intestazione per ricevere i dati di input richiesti dal client. Attiviamo il validatore di richieste API Gateway API per garantire che tutti i dati richiesti siano definiti e specificati correttamente. Configuriamo un modello di mappatura per API Gateway per trasformare i dati di richiesta forniti dal client nel formato valido richiesto dalla funzione Lambda di backend.

Per creare un file API con un'integrazione non proxy Lambda

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com /apigateway.
2. Se è la prima volta che utilizzi API Gateway, viene visualizzata una pagina che presenta le funzionalità del servizio. In RESTAPI, scegli Costruisci. Quando viene visualizzato il API popup Crea esempio, scegli OK.

   Se non è la prima volta che usi API Gateway, scegli Crea API. In RESTAPI, scegli Build.
3. Per APInome, inserisci**`LambdaNonProxyAPI`**.

4. (Facoltativo) In Description (Descrizione), immettere una descrizione.
5. Mantieni il tipo di API endpoint impostato su Regionale.
6. Scegli Crea API.

Dopo aver creato la tuaAPI, crei una risorsa/{city}. Questo è l'esempio di una risorsa con una variabile di percorso che riceve un input dal client. Più avanti mapperai questa variabile nell'input della funzione Lambda utilizzando un modello di mappatura.

Per creare una risorsa

1. Scegli Crea risorsa.
2. Mantieni l'opzione Risorsa proxy disattivata.
3. Mantieni Percorso risorsa impostato su `/`.
4. Per Resource Name (Nome risorsa) immetti **`{city}`**.
5. Mantieni CORS(Cross Origin Resource Sharing) disattivata.
6. Scegli Crea risorsa.

Dopo la risorsa /{city}, è necessario creare un metodo `ANY`. Il `ANY` HTTP verbo è un segnaposto per un HTTP metodo valido che un client invia in fase di esecuzione. Questo esempio mostra che il metodo `ANY` può essere utilizzato sia per l'integrazione Lambda personalizzata sia per quella proxy.

Per creare un metodo **`ANY`**

1. Seleziona la risorsa /{city}, quindi scegli Crea metodo.
2. In Tipo di metodo, seleziona ANY.
3. Per Tipo di integrazione seleziona Funzione Lambda.
4. Mantieni l'opzione Integrazione proxy Lambda disattivata.
5. Per la funzione Lambda, seleziona il Regione AWS luogo in cui hai creato la funzione Lambda, quindi inserisci il nome della funzione.
6. Scegli le impostazioni di richiesta del metodo.

   Ora, attivi un validatore di richiesta per una variabile di URL percorso, un parametro della stringa di query e un'intestazione per garantire che tutti i dati richiesti siano definiti. Per questo esempio, crea un parametro della stringa di query `time` e un'intestazione `day`.
7. Per Validatore richiesta seleziona Convalida parametri di stringa query e intestazioni.

8. Scegliete i parametri della stringa di URL query ed effettuate le seguenti operazioni:
   a. Scegliere Add query string (Aggiungi stringa di query).
   b. Per Nome, immetti **`time`**.
   c. Attiva Campo obbligatorio.
   d. Mantieni disattivata l'opzione Caching.
9. Scegliete le intestazioni della HTTP richiesta ed effettuate le seguenti operazioni:
   a. Seleziona Add header (Aggiungi intestazione).
   b. Per Nome, immetti **`day`**.
   c. Attiva Campo obbligatorio.
   d. Mantieni disattivata l'opzione Caching.
10. Scegli Crea metodo.

Dopo aver attivato un validatore di richieste, configuri la richiesta di integrazione per il `ANY` metodo aggiungendo un modello di body mapping per trasformare la richiesta in entrata in un JSON payload, come richiesto dalla funzione Lambda di backend.

Per configurare la richiesta di integrazione

1. Nella scheda Richiesta di integrazione, nelle impostazioni della richiesta di integrazione, scegli Modifica.
2. Per Richiesta corpo passthrough scegli Quando non ci sono modelli definiti (consigliato).
3. Scegli Modelli di mappatura.
4. Scegliere Add mapping template (Aggiungi modello di mappatura).
5. Per Tipo di contenuto inserisci **`application/json`**.
6. Per Corpo del modello inserisci il seguente codice:

```
#set($inputRoot = $input.path('$'))
{
  "city": "$input.params('city')",
  "time": "$input.params('time')",
  "day":  "$input.params('day')",
  "name": "$inputRoot.callerName"
}
```

7. Seleziona Salva.

Prova a richiamare il metodo API

La console API Gateway fornisce una struttura di test che consente di eseguire il test di invocazione API prima della distribuzione. È possibile utilizzare la funzionalità Test della console per eseguire il test API inviando la seguente richiesta:

```
POST /Seattle?time=morning
day:Wednesday

{
    "callerName": "John"
}
```

In questa richiesta di test imposterai `ANY` su `POST` e `{city}` su `Seattle` e assegnerai `Wednesday` come valore di intestazione di `day` e `"John"` come valore di `callerName`.

Test del metodo **`ANY`**

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. In Tipo di metodo, seleziona `POST`.
3. In Percorso immetti **`Seattle`** per city.
4. In Stringhe di query immetti **`time=morning`**.
5. In Intestazioni immetti **`day:Wednesday`**.
6. In Corpo della richiesta immetti **`{ "callerName": "John" }`**.
7. Scegli Test (Esegui test).

Verifica che il payload della risposta restituita sia come segue:

```
{
  "greeting": "Good morning, John of Seattle. Happy Wednesday!"
}
```

È inoltre possibile visualizzare i log per esaminare come API Gateway elabora la richiesta e la risposta.

```
Execution log for request test-request
Thu Aug 31 01:07:25 UTC 2017 : Starting execution for request: test-invoke-request
Thu Aug 31 01:07:25 UTC 2017 : HTTP Method: POST, Resource Path: /Seattle
Thu Aug 31 01:07:25 UTC 2017 : Method request path: {city=Seattle}
Thu Aug 31 01:07:25 UTC 2017 : Method request query string: {time=morning}
Thu Aug 31 01:07:25 UTC 2017 : Method request headers: {day=Wednesday}
Thu Aug 31 01:07:25 UTC 2017 : Method request body before transformations:
 { "callerName": "John" }
Thu Aug 31 01:07:25 UTC 2017 : Request validation succeeded for content type
 application/json
Thu Aug 31 01:07:25 UTC 2017 : Endpoint request URI: https://
lambda.us-west-2.amazonaws.com/2015-03-31/functions/arn:aws:lambda:us-
west-2:123456789012:function:GetStartedLambdaIntegration/invocations
Thu Aug 31 01:07:25 UTC 2017 : Endpoint request headers: {x-amzn-lambda-integration-
tag=test-request,
 Authorization=**********************************************************************
 X-Amz-Date=20170831T010725Z, x-amzn-apigateway-api-id=beags1mnid, X-Amz-
Source-Arn=arn:aws:execute-api:us-west-2:123456789012:beags1mnid/null/POST/
{city}, Accept=application/json, User-Agent=AmazonAPIGateway_beags1mnid,
 X-Amz-Security-Token=FQoDYXdzELL//////////wEaDMHGzEdEOT/VvGhabiK3AzgKrJw
+3zLqJZG4PhOq12K6W21+QotY2rrZyOzqhLoiuRg3CAYNQ2eqgL5D54+63ey9bIdtwHGoyBdq8ecWxJK/
YUnT2Rau0L9HCG5p7FC05h3IvwlFfvcidQNXeYvsKJTLXI05/
yEnY3ttIAnpNYLOezD9Es8rBfyruHfJfOqextKlsC8DymCcqlGkig8qLKcZ0hWJWVwiPJiFgL7laabXs+
+ZhCa4hdZo4iqlG729DE4gaV1mJVdoAagIUwLMo+y4NxFDu0r7I0/
EO5nYcCrppGVVBYiGk7H4T6sXuhTkbNNqVmXtV3ch5bOlh7 [TRUNCATED]
Thu Aug 31 01:07:25 UTC 2017 : Endpoint request body after transformations: {
  "city": "Seattle",
  "time": "morning",
  "day": "Wednesday",
  "name" : "John"
}
Thu Aug 31 01:07:25 UTC 2017 : Sending request to https://lambda.us-
west-2.amazonaws.com/2015-03-31/functions/arn:aws:lambda:us-
west-2:123456789012:function:GetStartedLambdaIntegration/invocations
Thu Aug 31 01:07:25 UTC 2017 : Received response. Integration latency: 328 ms
Thu Aug 31 01:07:25 UTC 2017 : Endpoint response body before transformations:
 {"greeting":"Good morning, John of Seattle. Happy Wednesday!"}
Thu Aug 31 01:07:25 UTC 2017 : Endpoint response headers: {x-amzn-Remapped-Content-
Length=0, x-amzn-RequestId=c0475a28-8de8-11e7-8d3f-4183da788f0f, Connection=keep-
alive, Content-Length=62, Date=Thu, 31 Aug 2017 01:07:25 GMT, X-Amzn-Trace-
Id=root=1-59a7614d-373151b01b0713127e646635;sampled=0, Content-Type=application/json}
Thu Aug 31 01:07:25 UTC 2017 : Method response body after transformations:
 {"greeting":"Good morning, John of Seattle. Happy Wednesday!"}
```

```
Thu Aug 31 01:07:25 UTC 2017 : Method response headers: {X-Amzn-Trace-
Id=sampled=0;root=1-59a7614d-373151b01b0713127e646635, Content-Type=application/json}
Thu Aug 31 01:07:25 UTC 2017 : Successfully completed execution
Thu Aug 31 01:07:25 UTC 2017 : Method completed with status: 200
```

I log mostrano la richiesta in ingresso prima della mappatura e la richiesta di integrazione dopo la mappatura. I log risultano utili per valutare, in caso di esito negativo di un test, se l'input originale è corretto o il modello di mappatura funziona correttamente.

Implementa il API

La chiamata di test è una simulazione e presenta alcune limitazioni. Ad esempio, ignora qualsiasi meccanismo di autorizzazione attivato su. API Per testare l'APIesecuzione in tempo reale, è necessario implementare il primo. API Per distribuire unAPI, è necessario creare una fase in cui creare un'istantanea del API momento. Il nome dello stage definisce anche il percorso di base dopo il nome API host predefinito. La API risorsa root viene aggiunta dopo il nome dello stage. Quando si modifica laAPI, è necessario ridistribuirla in una fase nuova o esistente prima che le modifiche abbiano effetto.

Per distribuirlo su uno stadio API

1. Scegliere Deploy (Implementa)API.
2. In Fase, seleziona Nuova fase.
3. In Stage name (Nome fase) immettere **`test`**.

   > **Note**
   > L'input deve essere testo codificato a UTF -8 (cioè non localizzato).

4. (Facoltativo) In Description (Descrizione), immettere una descrizione.
5. Seleziona Deploy (Implementa).

In Dettagli dello stage, scegli l'icona di copia per copiare la tua API chiamata. URL Lo schema generale della base API URL è`https://`*`api-id`*`.`*`region`*`.amazonaws.com/`*`stageName`*. Ad esempio, la base URL di API (`beags1mnid`) creata nella `us-west-2` regione e distribuita sullo `test` stage è`https://beags1mnid.execute-api.us-west-2.amazonaws.com/test`.

Testatela API in una fase di distribuzione

Esistono diversi modi per testare una distribuzioneAPI. Per GET le richieste che utilizzano solo variabili di URL percorso o parametri della stringa di query, è possibile inserire la API risorsa URL in un browser. Per altri metodi, è necessario utilizzare utilità di REST API test più avanzate, come POSTMANo c URL.

Per testare l'APIuso di c URL

1. Apri una finestra terminale sul tuo computer locale connesso a Internet.
2. Per testare `POST /Seattle?time=evening`:

   Copia il seguente URL comando c e incollalo nella finestra del terminale.

   ```
   curl -v -X POST \
     'https://beags1mnid.execute-api.us-west-2.amazonaws.com/test/Seattle?
   time=evening' \
     -H 'content-type: application/json' \
     -H 'day: Thursday' \
     -H 'x-amz-docs-region: us-west-2' \
     -d '{
    "callerName": "John"
   }'
   ```

   Dovresti visualizzare una risposta di operazione riuscita con il payload seguente:

   ```
   {"greeting":"Good evening, John of Seattle. Happy Thursday!"}
   ```

   Se modifichi `POST` in `PUT` in questa richiesta di metodo, otterrai la stessa risposta.

Elimini

Se non hai più bisogno delle funzioni Lambda create per questa procedura dettagliata, puoi eliminarle in questa fase. Puoi anche eliminare le IAM risorse di accompagnamento.

> **Warning**
>
> Se intendi completare le altre procedure guidate di questa serie, non eliminare il ruolo di esecuzione o di invocazione Lambda. Se elimini una funzione Lambda su APIs cui fai

affidamento, quella non APIs funzionerà più. L'eliminazione di una funzione Lambda non può essere annullata. Se desideri utilizzare di nuovo la funzione Lambda, dovrai ricrearla.
Se elimini una IAM risorsa su cui si basa una funzione Lambda, quella funzione Lambda non funzionerà più e tutte le risorse APIs che si basano su quella funzione non funzioneranno più. L'eliminazione di una IAM risorsa non può essere annullata. Se si desidera utilizzare nuovamente la IAM risorsa, è necessario ricrearla.

Per eliminare la funzione Lambda

1. Accedi a AWS Management Console e apri la AWS Lambda console all'indirizzo https://console.aws.amazon.com/lambda/.
2. Dall'elenco delle funzioni, scegli GetStartedLambdaIntegration, scegli Azioni, quindi scegli Elimina funzione. Quando viene richiesto, scegliere nuovamente Delete (Elimina).

Per eliminare le risorse IAM associate

1. Apri la IAM console all'indirizzo https://console.aws.amazon.com/iam/.
2. In Details (Dettagli) scegliere Roles (Ruoli).
3. Dall'elenco dei ruoli, scegli GetStartedLambdaIntegrationRole, scegli Azioni di ruolo, quindi scegli Elimina ruolo. Segui i passaggi indicati nella console per eliminare il ruolo.

Tutorial: Creazione di un'RESTAPIintegrazione proxy Lambda con più account

Ora puoi usare una AWS Lambda funzione di un altro AWS account come backend di API integrazione. Ogni account può trovarsi in qualsiasi regione in cui Amazon API Gateway è disponibile. Ciò semplifica la gestione e la condivisione centralizzata delle funzioni di backend Lambda tra più utenti. APIs

In questa sezione, mostriamo come configurare l'integrazione del proxy Lambda tra account utilizzando la console API Amazon Gateway.

Create API per l'integrazione Lambda tra più account di API Gateway

Per creare un API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.

2. Se è la prima volta che utilizzi API Gateway, viene visualizzata una pagina che presenta le funzionalità del servizio. In RESTAPI, scegli Costruisci. Quando viene visualizzato il API popup Crea esempio, scegli OK.

   Se non è la prima volta che usi API Gateway, scegli Crea API. In RESTAPI, scegli Build.
3. Per APInome, inserisci**`CrossAccountLambdaAPI`**.
4. (Facoltativo) In Description (Descrizione), immettere una descrizione.
5. Mantieni il tipo di API endpoint impostato su Regionale.
6. Scegli Crea API.

Creazione di una funzione di integrazione Lambda in un altro account

Ora creerai una funzione Lambda in un account diverso da quello in cui hai creato l'esempio. API

Creazione di una funzione Lambda in un altro account

1. Accedi alla console Lambda con un account diverso da quello in cui hai creato il tuo API Gateway. API
2. Selezionare Create function (Crea funzione).
3. Scegli Author from scratch (Crea da zero).
4. In Author from scratch (Crea da zero), effettuare le seguenti operazioni:

   a. Per Function name (Nome funzione) immettere un nome.
   b. Dall'elenco a discesa Runtime, scegliere un runtime Node.js supportato.
   c. Per Architecture, mantieni l'impostazione predefinita.
   d. In Autorizzazioni, espandere Scegli o crea un ruolo di esecuzione. È possibile creare un ruolo o scegliere un ruolo esistente.
   e. Selezionare Create function (Crea funzione) per continuare.
5. Scorrere verso il basso fino al riquadro Function code (Codice funzione).
6. Immetti l'implementazione della funzione Node.js da the section called “Tutorial: Creare un'integrazione REST API con un proxy Lambda”.
7. Seleziona Deploy (Implementa).
8. Nota la versione completa ARN della tua funzione (nell'angolo in alto a destra del riquadro delle funzioni Lambda). in quanto sarà necessario per la creazione dell'integrazione Lambda tra più account.

Configurazione dell'integrazione Lambda tra più account

Una volta che hai una funzione di integrazione Lambda in un altro account, puoi utilizzare la console API Gateway per aggiungerla API al tuo primo account.

> **Note**
>
> Se stai configurando un autorizzatore interregionale e tra più account, `sourceArn` quello che viene aggiunto alla funzione di destinazione deve utilizzare la regione della funzione, non l'area di. API

Dopo aver creato unAPI, crei una risorsa. In genere, API le risorse sono organizzate in un albero di risorse in base alla logica dell'applicazione. In questo esempio crei una risorsa /helloworld.

Per creare una risorsa

1. Scegli Crea risorsa.
2. Mantieni l'opzione Risorsa proxy disattivata.
3. Mantieni Percorso risorsa impostato su `/`.
4. Per Resource Name (Nome risorsa) immetti **`helloworld`**.
5. Mantieni CORS(Cross Origin Resource Sharing) disattivato.
6. Scegli Crea risorsa.

Dopo la risorsa, è necessario creare un metodo `GET`. Integra il metodo `GET` con una funzione Lambda in un altro account.

Creazione di un metodo **`GET`**

1. Seleziona la risorsa /helloworld, quindi scegli Crea metodo.
2. In Tipo di metodo, seleziona GET.
3. Per Tipo di integrazione seleziona Funzione Lambda.
4. Attiva l'opzione Integrazione proxy Lambda.
5. Per la funzione Lambda, inserisci l'intera ARN funzione Lambda dal passaggio 1.

   Nella console Lambda, puoi trovare la funzione ARN per la tua funzione nell'angolo in alto a destra della finestra della console.

6. Quando si immetteARN, viene visualizzata una stringa di `aws lambda add-permission` comando. Questa policy fornisce al tuo primo account l'accesso alla funzione Lambda del secondo account. Copia e incolla la stringa di `aws lambda add-permission` comando in una AWS CLI finestra configurata per il tuo secondo account.
7. Scegli Crea metodo.

Puoi visualizzare la policy aggiornata per la funzione nella console Lambda.

(Facoltativo) Per visualizzare la policy aggiornata

1. Accedi a AWS Management Console e apri la AWS Lambda console all'indirizzo https://console.aws.amazon.com/lambda/.
2. Seleziona la funzione Lambda.
3. Seleziona Autorizzazioni.

   Dovresti vedere una `Allow` policy con una `Condition` clausola in cui in the `AWS:SourceArn` corrisponde ARN al tuo API `GET` metodo.

Tutorial: creazione di un'API REST mediante l'importazione di un esempio

Puoi utilizzare la console Amazon API Gateway per creare e testare una semplice API REST con l'integrazione HTTP per un PetStore sito Web. La definizione dell'API è preconfigurata come file OpenAPI 2.0. Dopo aver caricato la definizione dell'API in API Gateway, puoi usare la console API Gateway per esaminare la struttura di base dell'API o semplicemente distribuire e testare l'API.

L'API di PetStore esempio supporta i seguenti metodi per consentire a un client di accedere al sito Web di backend HTTP di`http://petstore-demo-endpoint.execute-api.com/petstore/pets`.

> **Note**
>
> Questo tutorial utilizza un endpoint HTTP come esempio. Quando crei le tue API, consigliamo di utilizzare gli endpoint HTTPS per le integrazioni HTTP.

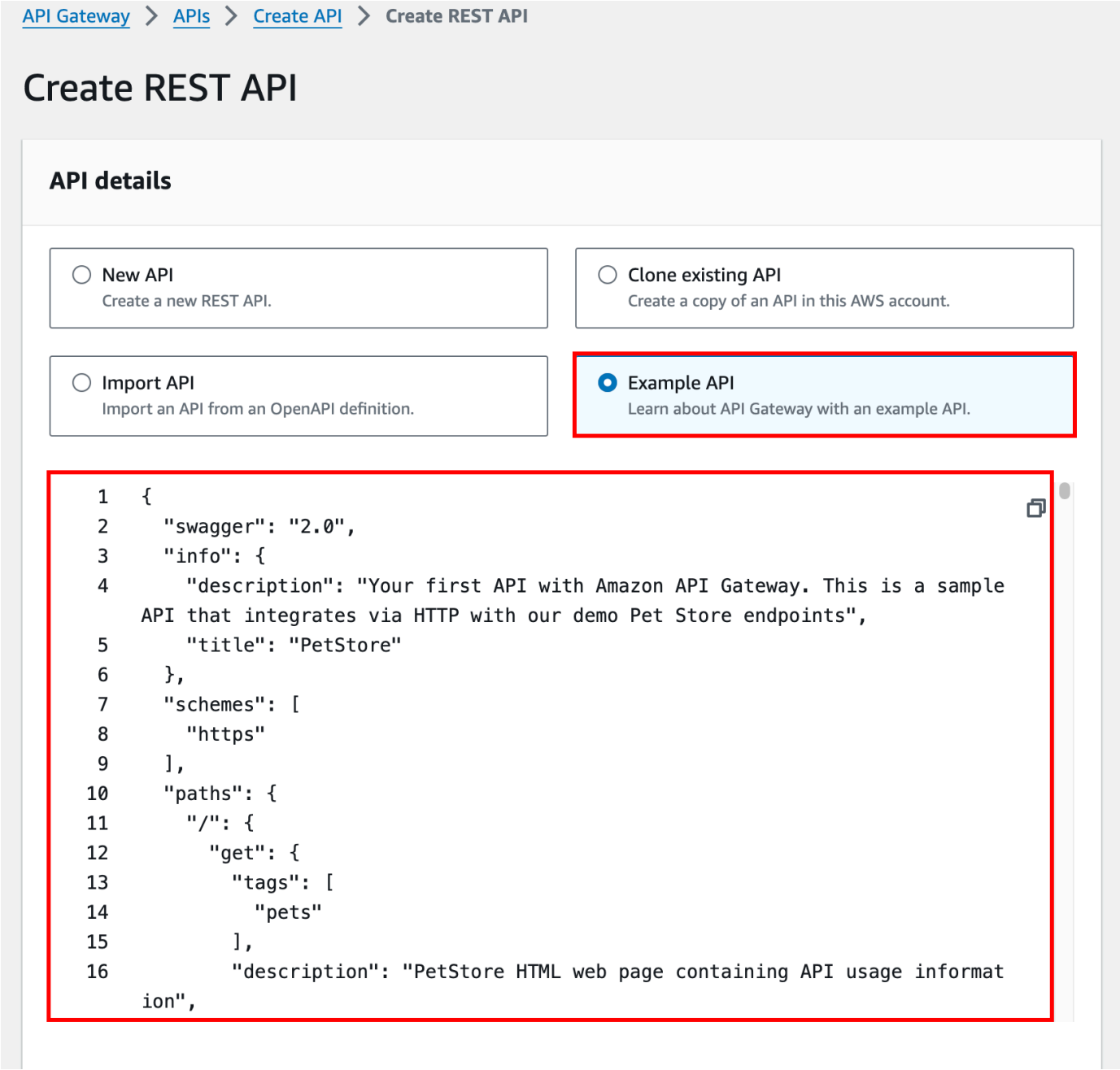
- `GET /`: per l'accesso in lettura della risorsa radice dell'API non integrata con un endpoint di back-end. API Gateway risponde con una panoramica del PetStore sito Web. Questo è un esempio del tipo di integrazione `MOCK`.
- `GET /pets`: per l'accesso in lettura alla risorsa `/pets` dell'API integrata con la risorsa `/pets` di back-end omonima. Il backend restituisce una pagina di animali domestici disponibili in. PetStore Questo è un esempio del tipo di integrazione `HTTP`. L'URL dell'endpoint di integrazione è `http://petstore-demo-endpoint.execute-api.com/petstore/pets`.
- `POST /pets`: per l'accesso in scrittura alla risorsa `/pets` dell'API integrata con la risorsa `/petstore/pets` di back-end. Dopo aver ricevuto una richiesta corretta, il backend aggiunge l'animale domestico specificato a PetStore e restituisce il risultato al chiamante. L'integrazione è anche `HTTP`.
- `GET /pets/{petId}`: per l'accesso in lettura a un animale domestico identificato da un valore `petId` specificato come variabile di percorso dell'URL di richiesta in entrata. Questo metodo ha anche il tipo di integrazione `HTTP`. Il backend restituisce l'animale domestico specificato trovato in. PetStore L'URL dell'endpoint HTTP di back-end è `http://petstore-demo-endpoint.execute-api.com/petstore/pets/`*n*, dove n è un valore intero come l'identificativo dell'animale domestico su cui è stata eseguita la query.

L'API supporta l'accesso CORS mediante i metodi `OPTIONS` del tipo di integrazione `MOCK`. API Gateway restituisce le intestazioni richieste che supportano l'accesso a CORS.

La procedura seguente descrive le diverse fasi per creare e testare un'API da un esempio mediante la console API Gateway.

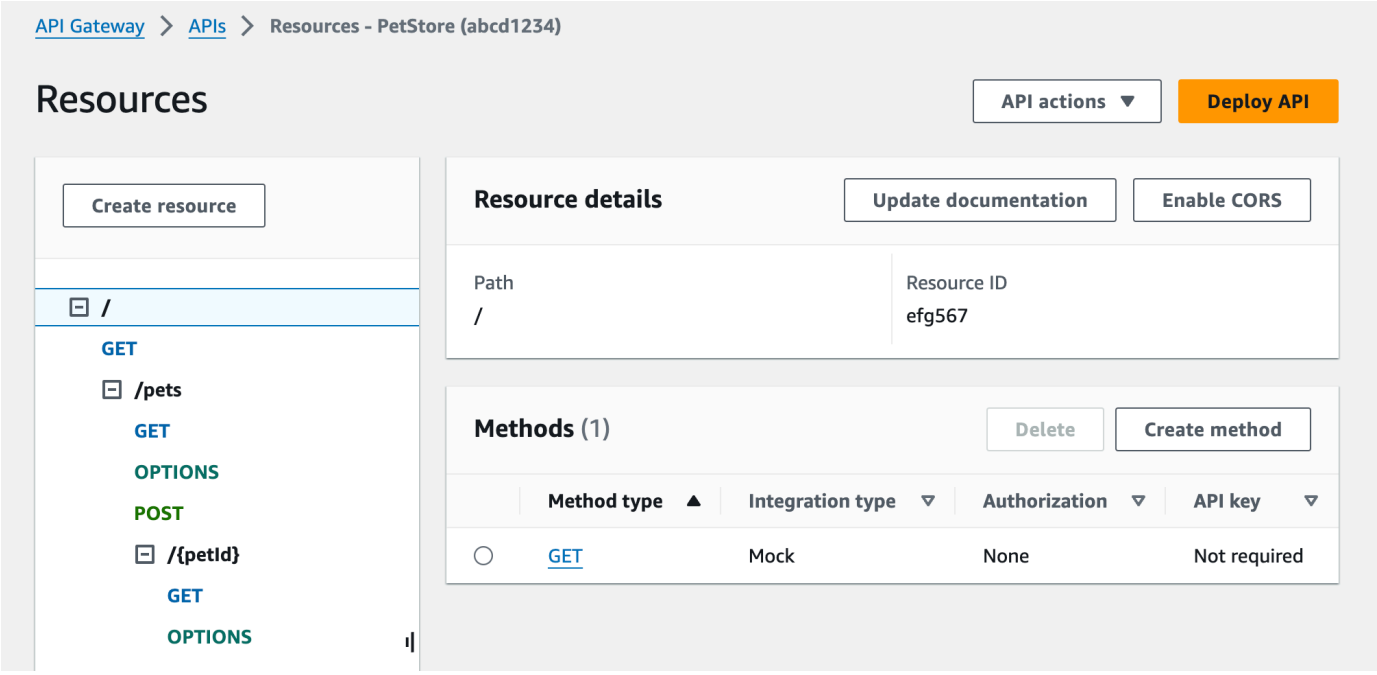
Per importare, creare ed eseguire il test dell'API di esempio

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Esegui una di queste operazioni:
   - Per creare la tua prima API, in REST API, scegli Crea.
   - Se hai già creato un'API, scegli Crea API, quindi scegliere Crea per API REST.
3. In Crea REST API scegli API di esempio, quindi seleziona Crea API per creare l'API di esempio.



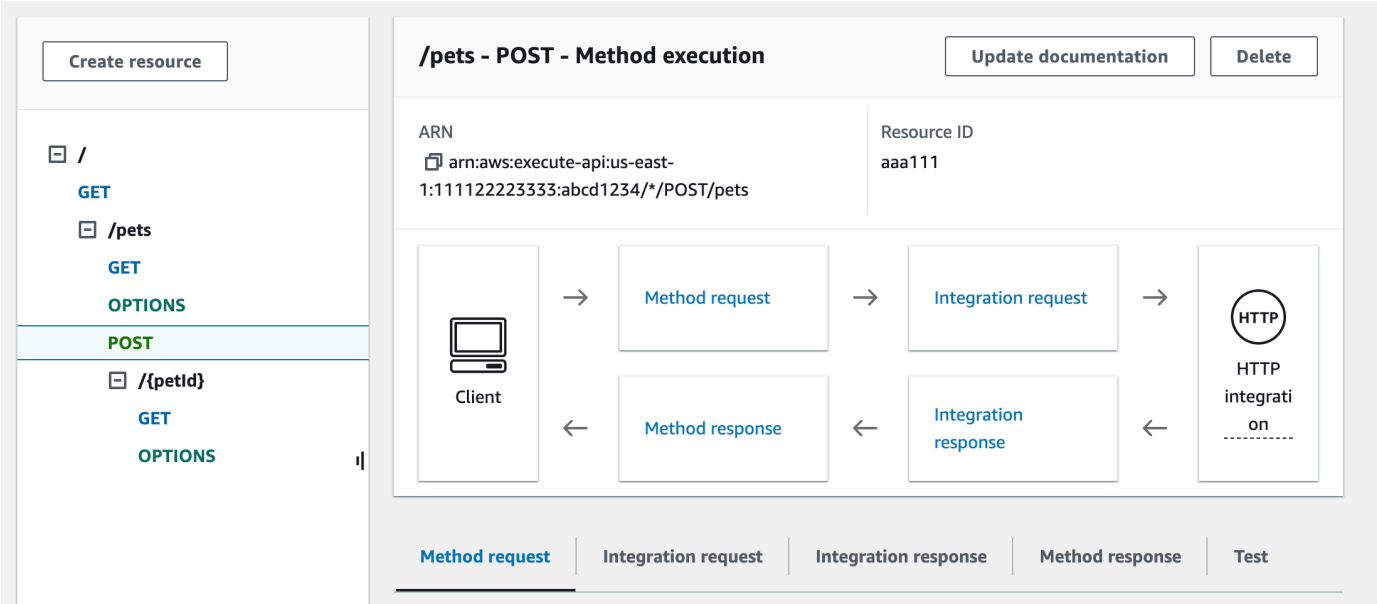
È possibile scorrere la definizione OpenAPI per i dettagli relativi a questa API di esempio prima di scegliere Crea API.

4. Nel riquadro di navigazione principale scegli Risorse. L'API appena creata viene mostrata come segue:



Il riquadro Resources (Risorse) mostra la struttura dell'API creata come albero di nodi. I metodi API definiti in ogni risorsa sono le estremità dell'albero. Quando una risorsa viene selezionata, tutti i relativi metodi vengono elencati nella tabella Metodi a destra. Con ogni metodo vengono visualizzati il tipo di metodo, il tipo di integrazione, il tipo di autorizzazione e il requisito della chiave API.

5. Per visualizzare i dettagli di un metodo, modificarne la configurazione o testarne la chiamata, scegli il nome del metodo nell'elenco o nella struttura di risorse. In questo caso scegliamo il metodo `POST /pets` a scopo illustrativo:



Il riquadro Esecuzione metodo risultante presenta una vista logica della struttura e dei comportamenti del metodo scelto (`POST /pets`).

La Richiesta metodo e la Risposta metodo rappresentano l'interfaccia dell'API con il front-end, mentre la Richiesta di integrazione e la Risposta di integrazione rappresentano l'interfaccia dell'API con il back-end.

Un client usa l'API per accedere a una funzionalità di back-end tramite la Richiesta metodo. Gateway API converte la richiesta del client, se necessario, nel formato accettabile per il back-end nella Richiesta di integrazione, prima di inoltrare la richiesta in entrata al back-end. La richiesta trasformata è nota come richiesta di integrazione. Analogamente, il back-end restituisce la risposta a Gateway API nella Risposta di integrazione. API Gateway quindi la instrada a Method Response (Risposta metodo) prima di inviarla al client. Anche in questo caso, se necessario, API Gateway può mappare i dati della risposta di back-end a un formato previsto dal client.

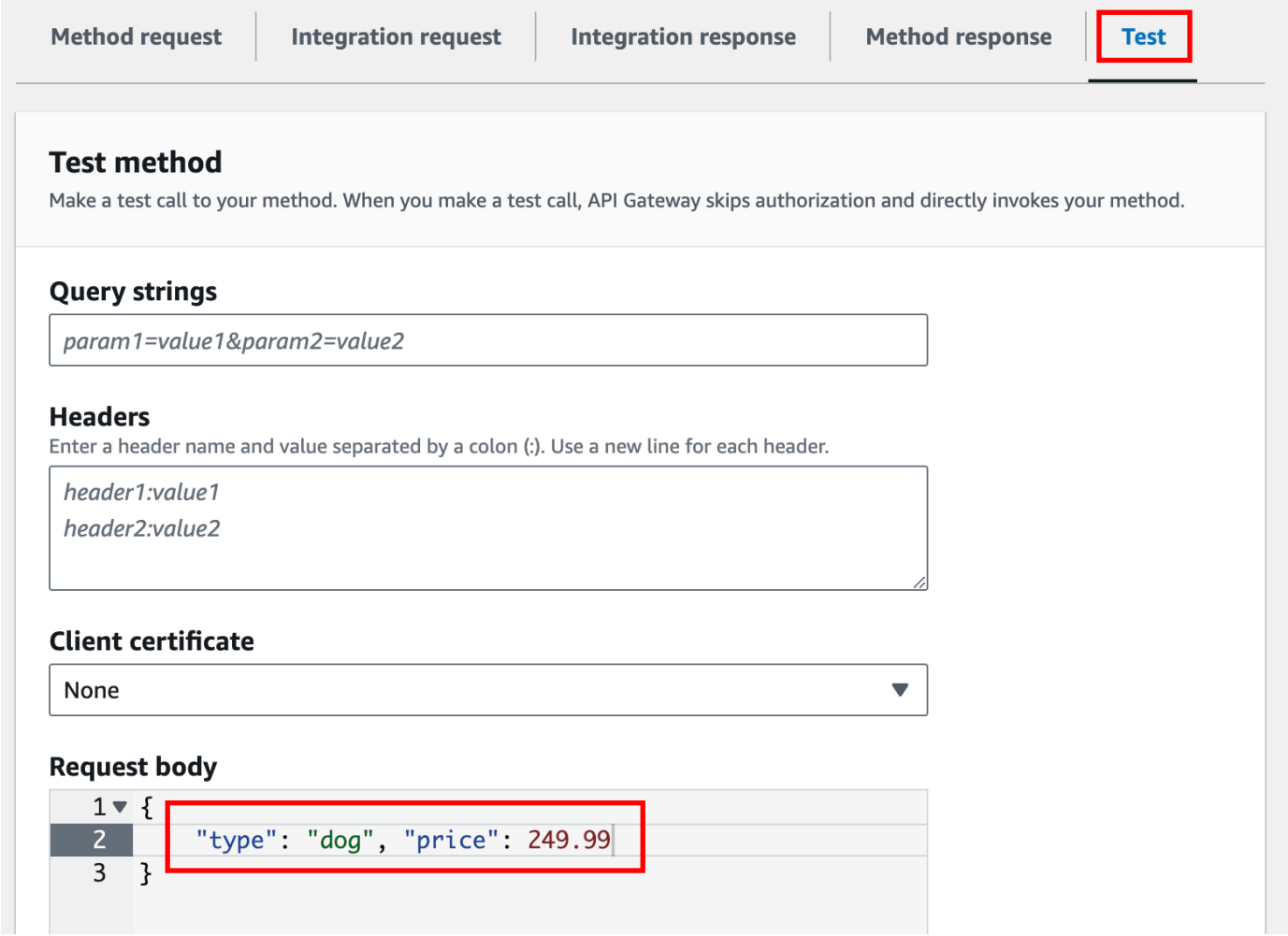
Per il metodo `POST` su una risorsa API, il payload della richiesta del metodo può essere passato attraverso la richiesta di integrazione senza alcuna modifica, se il formato del payload della richiesta del metodo è uguale a quello del payload della richiesta di integrazione.

La richiesta del metodo `GET /` usa il tipo di integrazione `MOCK` e non è legata a un endpoint di back-end reale. La Risposta di integrazione corrispondente è configurata per restituire una pagina HTML statica. Quando viene chiamato il metodo, Gateway API accetta la richiesta e restituisce immediatamente al client la risposta di integrazione configurata mediante la Risposta

metodo. Puoi usare l'integrazione fittizia per testare un'API senza richiedere un endpoint di back-end. Puoi usarla anche per fornire una risposta locale, generata da un modello di mappatura del corpo della risposta.

In qualità di sviluppatore dell'API, puoi controllare i comportamenti delle interazioni front-end dell'API mediante la configurazione della richiesta e della risposta di metodo. Puoi inoltre controllare i comportamenti delle interazioni back-end dell'API mediante la configurazione della richiesta e della risposta di integrazione. Ciò prevede le mappature dei dati tra un metodo e l'integrazione corrispondente. Per ora, ci concentriamo sul test dell'API per fornire un'esperienza end-to-end utente.

6. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
7. Ad esempio, per testare il metodo `POST /pets`, immetti il payload **`{"type": "dog","price": 249.99}`** seguente nel Corpo della richiesta prima di selezionare Test.



L'input specifica gli attributi dell'animale da aggiungere all'elenco degli animali domestici sul PetStore sito web.

8. I risultati visualizzati sono simili ai seguenti:

/pets - POST method test results

Request

/pets

Status

200

Latency

9

Response body

```
{
  "pet": {
    "type": "dog",
    "price": 249.99
  },
  "message": "success"
}
```

Response headers

```
{
  "Access-Control-Allow-Origin": "*",
  "Content-Type": "application/json",
  "X-Amzn-Trace-Id": "Root=1-65df8d2b-782cd3c572391cf4a85295f5"
}
```

Log

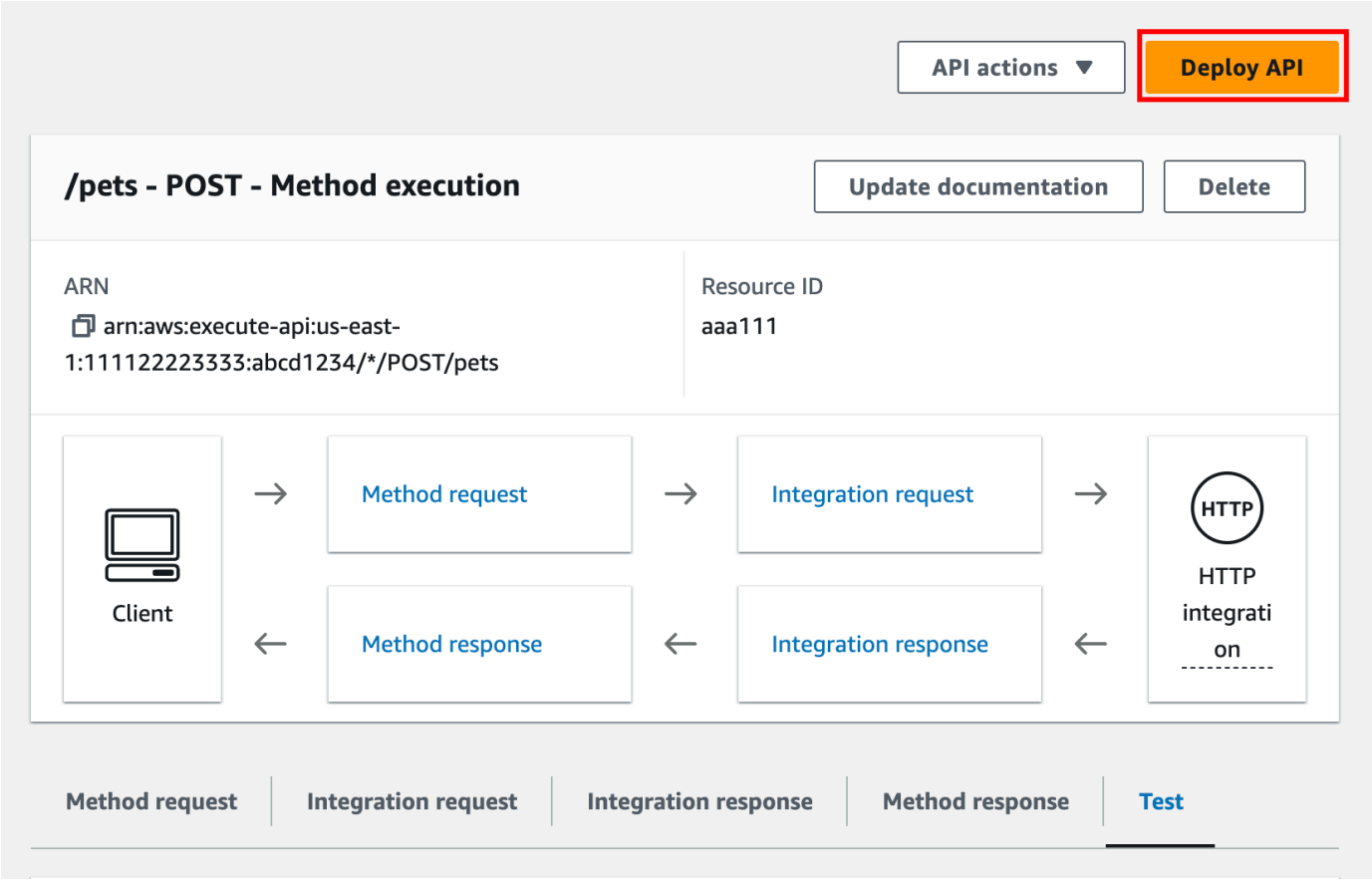
```
Execution log for request 30f01060-307f-4447-803c-61679ea4c5d6
Wed Feb 28 19:44:43 UTC 2024 : Starting execution for request: 30f01060-
307f-4447-803c-61679ea4c5d6
```

La voce Log dell'output mostra le modifiche dello stato dalla richiesta di metodo alla richiesta di integrazione e dalla risposta di integrazione alla risposta di metodo. Può risultare utile per la risoluzione dei problemi legati agli errori di mappatura che impediscono la riuscita della richiesta. In questo esempio non viene applicata alcuna mappatura: il payload della richiesta di metodo

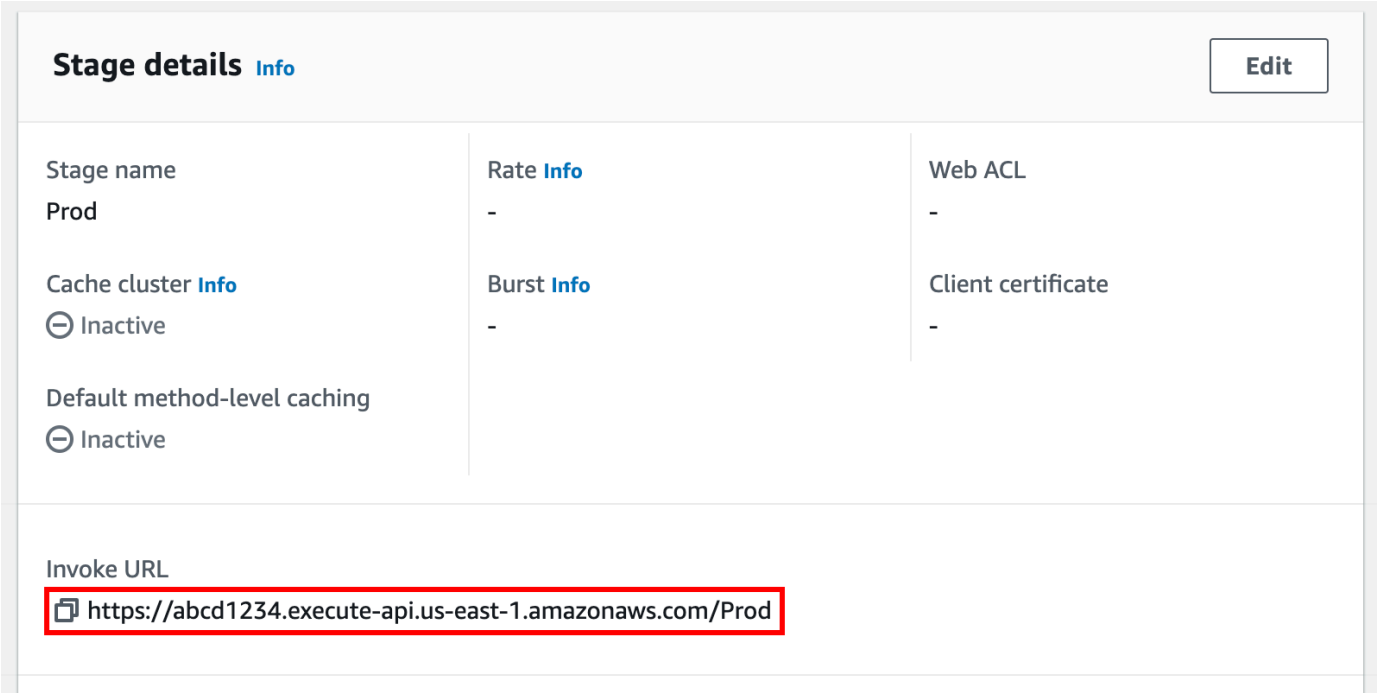
viene passato al back-end attraverso la richiesta di integrazione e, analogamente, la risposta del back-end viene passata alla risposta del metodo attraverso la risposta di integrazione.

Per testare l'API utilizzando un client diverso dalla test-invoke-request funzionalità API Gateway, devi prima implementare l'API in una fase.

9. Seleziona Implementa API per implementare l'API di esempio.



10. Per Fase seleziona Nuova fase, quindi immetti **`test`**.
11. (Facoltativo) In Description (Descrizione), immettere una descrizione.
12. Seleziona Deploy (Implementa).
13. Nel riquadro Fasi risultante, il campo Richiama URL in Dettagli fase visualizza l'URL per richiamare la richiesta del metodo `GET /` dell'API.



14. Scegli l'icona Copia per copiare l'URL di richiamo dell'API, quindi immetti l'URL di richiamo dell'API in un browser Web. La risposta con esito positivo restituisce il risultato, generato dal modello di mappatura nella risposta di integrazione.
15. Nel pannello di navigazione Stages (Fasi), espandere la fase test, selezionare GET per `/pets/{petId}`, quindi copiare il valore Invoke URL (URL chiamata) di `https://`*`api-id`*`.execute-api.`*`region`*`.amazonaws.com/test/pets/{petId}`. `{petId}` indica una variabile di percorso.

    Incollare il valore Invoke URL (URL chiamata) ottenuto nella fase precedente nella barra degli indirizzi di un browser, sostituendo `{petId}` con, ad esempio, `1` e premere Invio per inviare la richiesta. Dovrebbe essere restituita una risposta 200 OK con il payload JSON seguente:

    ```
    {
      "id": 1,
      "type": "dog",
      "price": 249.99
    }
    ```

    L'invocazione del metodo API nel modo indicato è possibile in quanto il tipo di Authorization (Autorizzazione) è impostato su `NONE`. Se venisse utilizzata l'autorizzazione `AWS_IAM`, la richiesta verrebbe firmata con i protocolli Signature Version 4 (SigV4). Per un esempio di questo

tipo di richiesta, consulta the section called “Tutorial: Crea un file REST API con un'integrazione HTTP non proxy”.

Scegli un tutorial sull'integrazione HTTP

Per creare un'API con integrazione HTTP, puoi utilizzare un'integrazione proxy HTTP o un'integrazione HTTP personalizzata.

In un'integrazione con proxy HTTP, è sufficiente impostare il metodo HTTP e l'URI dell'endpoint HTTP, in base ai requisiti del backend. Ti consigliamo di utilizzare l'integrazione del proxy HTTP, quando possibile, per sfruttare la configurazione semplificata dell'API.

Potresti voler utilizzare un'integrazione HTTP personalizzata se devi trasformare i dati delle richieste del client per il backend o trasformare i dati di risposta del backend per il client.

Argomenti

- Tutorial: creare un'API REST con un'integrazione proxy HTTP
- Tutorial: creare un file REST API con un'HTTPintegrazione non proxy

Tutorial: creare un'API REST con un'integrazione proxy HTTP

L'integrazione proxy HTTP è un meccanismo semplice, potente e versatile per creare un'API che permetta a un'applicazione Web di accedere a più caratteristiche o risorse dell'endpoint HTTP integrato, ad esempio l'intero sito Web, con una configurazione semplificata di un solo metodo API. In un'integrazione proxy HTTP, API Gateway passa la richiesta di metodo inviata dal client al back-end. I dati di richiesta che vengono passati includono intestazioni di richiesta, parametri delle stringhe di query, variabili di percorso URL e payload. L'endpoint HTTP di back-end o il server Web analizza i dati delle richieste in ingresso per stabilire la risposta da restituire. L'integrazione proxy HTTP consente l'interazione diretta tra client e back-end senza alcun intervento da parte di API Gateway dopo la configurazione del metodo API, tranne nel caso di problemi noti elencati in , ad esempio caratteri non supportat the section called “Note importanti”.

Con la risorsa proxy globale `{proxy+}` e il verbo catch-all `ANY` per il metodo HTTP, puoi usare un'integrazione proxy HTTP per creare un'API di un metodo API singolo. Il metodo espone l'intero set di operazioni e risorse HTTP accessibili pubblicamente di un sito Web. Quando il server Web di back-end apre più risorse per l'accesso pubblico, il client può usare le nuove risorse con la stessa configurazione API. Per consentirlo, lo sviluppatore del sito Web deve comunicare chiaramente con

lo sviluppatore del client in merito alla natura delle nuove risorse e alle operazioni applicabili per ciascuna di esse.

Come breve introduzione, il tutorial che segue fornisce un esempio di integrazione proxy HTTP. Nel tutorial, creiamo un'API utilizzando la console API Gateway per l'integrazione con il PetStore sito Web tramite una risorsa `{proxy+}` proxy generica e creiamo il segnaposto del metodo HTTP di. `ANY`

Argomenti

- Creazione di un'API con l'integrazione proxy HTTP mediante la console API Gateway
- Verifica di un'API con l'integrazione proxy HTTP

Creazione di un'API con l'integrazione proxy HTTP mediante la console API Gateway

Nella procedura seguente vengono descritte le fasi necessarie per creare e testare un'API con una risorsa proxy per un back-end HTTP mediante la console API Gateway. Il back-end HTTP è il sito Web `PetStore` (`http://petstore-demo-endpoint.execute-api.com/petstore/pets`) da Tutorial: creare un file REST API con un'HTTPintegrazione non proxy, in cui vengono usati screenshot come supporto visivo per illustrare gli elementi dell'interfaccia utente API Gateway. Se non hai esperienza di utilizzo della console API Gateway per la creazione di un'API, può essere di aiuto seguire prima tale sezione.

Per creare un API

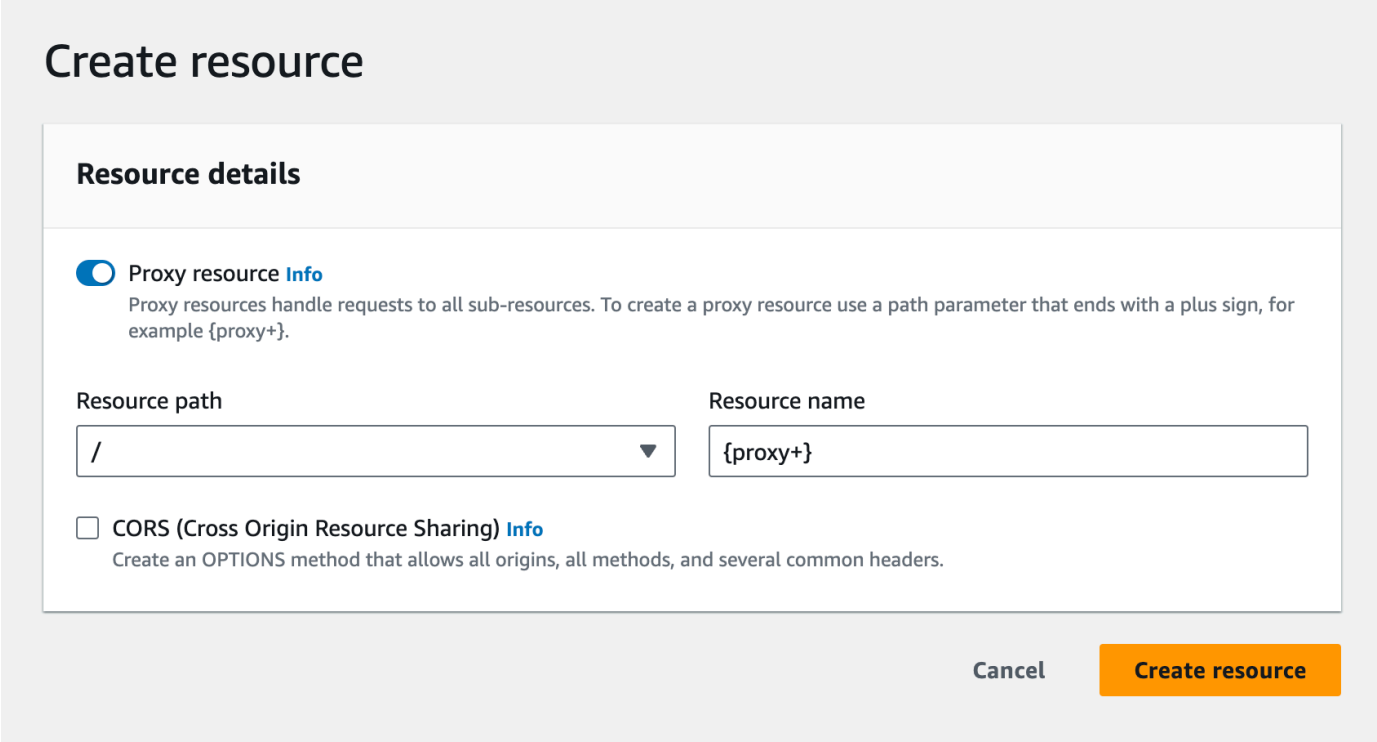
1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Se si utilizza API Gateway per la prima volta, verrà visualizzata una pagina che presenta le caratteristiche del servizio. In API REST, scegliere Crea. Quando appare il popup Create Example API (Crea API di esempio), scegliere OK.

   Se non è la prima volta che si utilizza API Gateway, scegliere Create API (Crea API). In API REST, scegliere Crea.
3. Per API name (Nome API), immettere **`HTTPProxyAPI`**.
4. (Facoltativo) In Description (Descrizione), immettere una descrizione.
5. Lasciare l'opzione Tipo di endpoint API impostata su Regionale.
6. Seleziona Create API (Crea API).

In questa fase crei un percorso della risorsa proxy di `{proxy+}`. Questo è il segnaposto di un endpoint di back-end in `http://petstore-demo-endpoint.execute-api.com/`. Ad esempio, può essere `petstore`, `petstore/pets` e `petstore/pets/{petId}`. Quando crei la risorsa `{proxy+}`, Gateway API crea il metodo `ANY` che funge da segnaposto per qualsiasi verbo HTTP supportato al runtime.

Per creare una risorsa /{proxy+}

1. Scegliere l'API.
2. Nel riquadro di navigazione principale scegli Risorse.
3. Scegli Crea risorsa.
4. Attiva Risorsa proxy.
5. Mantieni Percorso risorsa impostato su `/`.
6. Per Resource Name (Nome risorsa) immetti **`{proxy+}`**.
7. Mantieni CORS (Cross Origin Resource Sharing) disattivato.
8. Scegli Crea risorsa.



In questa fase integri il metodo `ANY` con un endpoint HTTP di back-end, utilizzando un'integrazione proxy. In un'integrazione proxy, Gateway API passa la richiesta del metodo inviata dal client al back-end senza intervento di Gateway API.

Per creare un metodo **`ANY`**

1. Scegli la risorsa /{proxy+}.
2. Scegli il metodo ANY.
3. Sotto il simbolo di avviso, scegli Modifica integrazione. Non puoi implementare un'API con un metodo senza un'integrazione.
4. Per Tipo di integrazione seleziona HTTP.
5. Attiva l'opzione Integrazione proxy HTTP.
6. Per Metodo HTTP seleziona ANY.
7. Per URL dell'endpoint immetti **`http://petstore-demo-endpoint.execute-api.com/{proxy}`**.
8. Selezionare Salva.

Verifica di un'API con l'integrazione proxy HTTP

L'esito di una richiesta client specifica dipende dagli elementi seguenti:

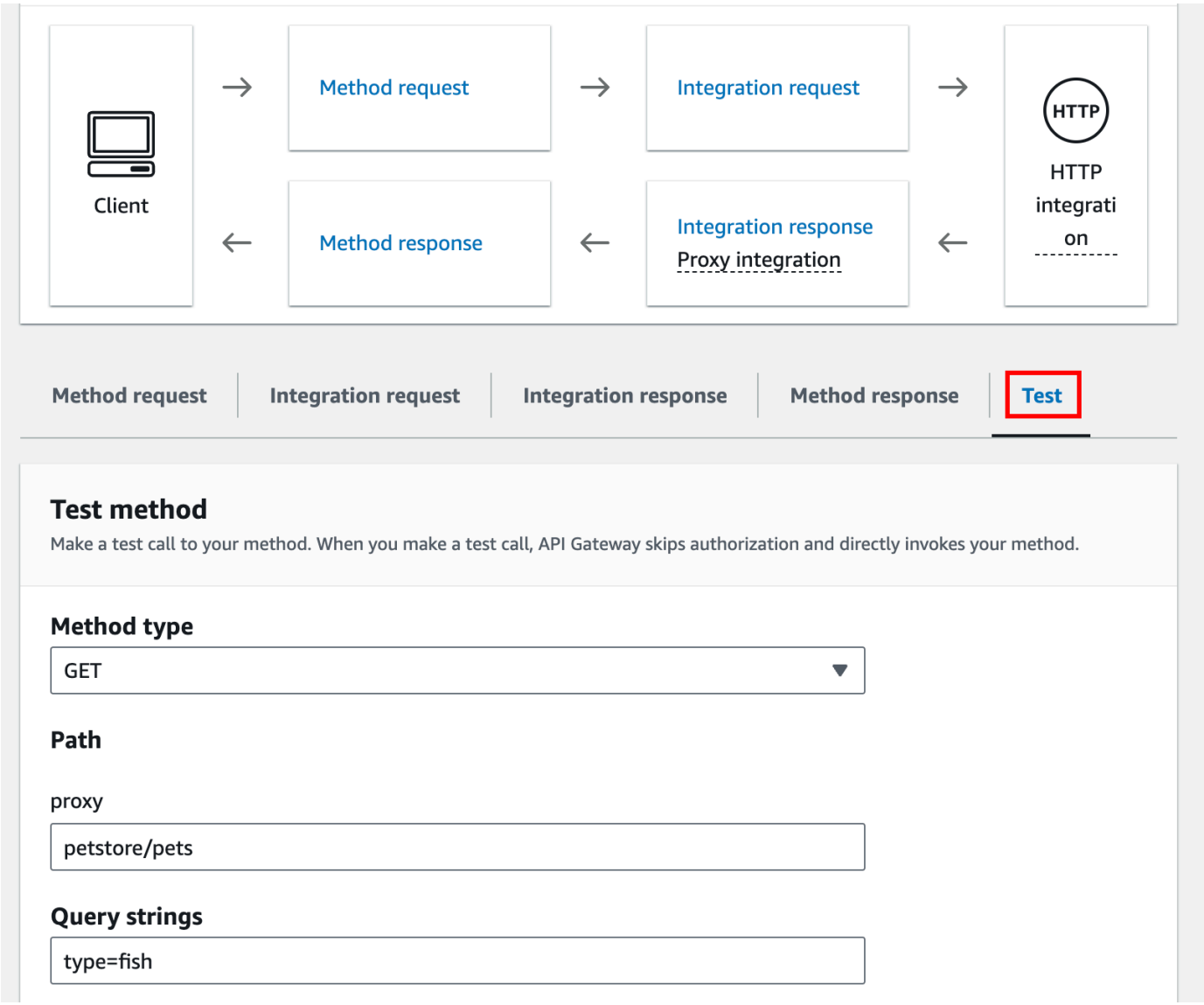
- Se il back-end ha reso disponibile l'endpoint di back-end corrispondente e, in caso affermativo, se ha concesso le autorizzazioni di accesso richieste.
- Se il client fornisce l'input corretto.

Ad esempio, l' PetStore API utilizzata qui non espone la risorsa. `/petstore` In questo modo, ottieni una risposta `404 Resource Not Found` contenente il messaggio di errore `Cannot GET /petstore`.

Inoltre per analizzare correttamente il risultato, il client deve essere in grado di gestire il formato di output del back-end. API Gateway non svolge alcuna mediazione per semplificare le interazioni tra il client e il back-end.

Per testare un'API integrata con il PetStore sito Web utilizzando l'integrazione del proxy HTTP tramite la risorsa proxy

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. In Tipo di metodo, seleziona `GET`.
3. In Percorso immetti **`petstore/pets`** per proxy.
4. In Stringhe di query immetti **`type=fish`**.
5. Scegli Test (Esegui test).



Poiché supporta la richiesta `GET /petstore/pets?type=fish`, il sito Web di back-end restituisce una risposta con esito positivo simile alla seguente:

```
[
  {
    "id": 1,
    "type": "fish",
    "price": 249.99
  },
  {
    "id": 2,
    "type": "fish",
    "price": 124.99
  },
  {
    "id": 3,
    "type": "fish",
    "price": 0.99
  }
]
```

Se provi a chiamare `GET /petstore`, ottieni una risposta `404` con il messaggio di errore `Cannot GET /petstore`. Questo perché il back-end non supporta l'operazione specificata. Se chiami`GET /petstore/pets/1`, riceverai una `200 OK` risposta con il seguente payload, poiché la richiesta è supportata dal PetStore sito Web.

```
{
  "id": 1,
  "type": "dog",
  "price": 249.99
}
```

Puoi anche utilizzare un browser per testare l'API. Implementa l'API e associala a una fase per creare l'URL di richiamata dell'API.

Per distribuire l'API

1. Seleziona Deploy API (Distribuisci API).
2. In Fase, seleziona Nuova fase.

3. In Stage name (Nome fase) immettere **`test`**.
4. (Facoltativo) In Description (Descrizione), immettere una descrizione.
5. Seleziona Deploy (Implementa).

Ora i client possono chiamare la tua API.

Richiamo dell'API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere l'API.
3. Nel pannello di navigazione principale scegli Fase.
4. In Dettagli fase, scegli l'icona Copia per copiare l'URL di richiamo dell'API.

   Immetti l'URL di richiamata dell'API in un browser Web.

   L'URL completo dovrebbe essere del tipo `https://abcdef123.execute-api.us-east-2.amazonaws.com/test/petstore/pets?type=fish`.

   Il tuo browser invia una richiesta `GET` all'API.
5. Il risultato sarà uguale a quello che viene restituito quando si usa Test nella console Gateway API.

Tutorial: creare un file REST API con un'HTTPintegrazione non proxy

In questo tutorial, ne crei una partendo API da zero utilizzando la console Amazon API Gateway. Puoi pensare alla console come a uno studio di API progettazione e utilizzarla per definire le API funzionalità, sperimentarne i comportamenti, crearla e distribuirla gradualmenteAPI. API

Argomenti

- Crea un file API con integrazione personalizzata HTTP
- (Facoltativo) Mappatura dei parametri di richiesta

Crea un file API con integrazione personalizzata HTTP

Questa sezione illustra i passaggi per creare risorse, esporre metodi su una risorsa, configurare un metodo per ottenere API i comportamenti desiderati e testare e implementare il. API

In questo passaggio, crei un file vuoto. API Nei passaggi seguenti creerai risorse e metodi per connetterti API all'`http://petstore-demo-endpoint.execute-api.com/petstore/pets`endpoint, utilizzando un'integrazione non proxyHTTP.

Per creare un API

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Se è la prima volta che utilizzi API Gateway, viene visualizzata una pagina che illustra le funzionalità del servizio. In RESTAPI, scegli Costruisci. Quando viene visualizzato il API popup Crea esempio, scegli OK.

   Se non è la prima volta che usi API Gateway, scegli Crea API. In RESTAPI, scegli Build.
3. Per APInome, inserisci**`HTTPNonProxyAPI`**.
4. (Facoltativo) In Description (Descrizione), immettere una descrizione.
5. Mantieni il tipo di API endpoint impostato su Regionale.
6. Scegli Crea API.

L'albero Resources (Risorse) mostra la risorsa root (`/`) senza metodi. In questo esercizio, lo creeremo API con l'integrazione HTTP personalizzata del PetStore sito Web (http://petstore-demo-endpoint.execute-api.com/petstore/pets.) A scopo illustrativo, creeremo una `/pets` risorsa come elemento secondario della radice e mostreremo su questa risorsa un GET metodo per consentire a un cliente di recuperare un elenco di articoli Pets disponibili dal sito Web. PetStore

Per creare una risorsa /pets

1. Scegli Crea risorsa.
2. Mantieni l'opzione Risorsa proxy disattivata.
3. Mantieni Percorso risorsa impostato su `/`.
4. Per Resource Name (Nome risorsa) immetti **`pets`**.
5. Mantieni CORS(Cross Origin Resource Sharing) disattivato.
6. Scegli Crea risorsa.

In questa fase crei un metodo `GET` per la risorsa /pets. Il metodo `GET` è integrato con il sito Web `http://petstore-demo-endpoint.execute-api.com/petstore/pets`. Altre opzioni per un API metodo includono le seguenti:

- POST, utilizzato principalmente per creare risorse per bambini.
- PUT, utilizzato principalmente per aggiornare le risorse esistenti (e, sebbene non consigliato, può essere utilizzato per creare risorse secondarie).
- DELETE, utilizzato per eliminare risorse.
- PATCH, utilizzato per aggiornare le risorse.
- HEAD, utilizzato principalmente negli scenari di test. È uguale GET ma non restituisce la rappresentazione delle risorse.
- OPTIONS, che può essere utilizzato dai chiamanti per ottenere informazioni sulle opzioni di comunicazione disponibili per il servizio di destinazione.

Per il HTTPmetodo della richiesta di integrazione, è necessario sceglierne uno supportato dal backend. Per `HTTP` o`Mock integration`, è logico che la richiesta del metodo e la richiesta di integrazione utilizzino lo stesso HTTP verbo. Per altri tipi di integrazione, la richiesta del metodo utilizzerà probabilmente un HTTP verbo diverso dalla richiesta di integrazione. Ad esempio, per chiamare una funzione Lambda, la richiesta di integrazione deve utilizzare `POST` per richiamare la funzione, mentre la richiesta del metodo può utilizzare qualsiasi HTTP verbo a seconda della logica della funzione Lambda.

Per creare un metodo **`GET`** nella risorsa /pets

1. Seleziona la risorsa /pets.
2. Scegli Crea metodo.
3. In Tipo di metodo, seleziona GET.
4. Per Tipo di integrazione, seleziona integrazione. HTTP
5. Mantieni disattivata l'integrazione del HTTP proxy.
6. Per HTTPmetodo, seleziona GET.
7. Per Endpoint URL, immettere**`http://petstore-demo-endpoint.execute-api.com/petstore/pets`**.

   Il PetStore sito Web consente di recuperare un elenco di `Pet` elementi in base al tipo di animale domestico, ad esempio «Cane» o «Gatto», in una determinata pagina.
8. Per Gestione contenuti seleziona Transito.
9. Scegli i parametri URL della stringa di interrogazione.

Il PetStore sito Web utilizza i parametri `type` e della stringa di `page` query per accettare un input. Si aggiungono i parametri della stringa di query alla richiesta del metodo e li si mappa nei parametri della stringa di query corrispondenti della richiesta di integrazione.

10. Per aggiungere i parametri della stringa di query, effettuate le seguenti operazioni:

    a. Scegliere Add query string (Aggiungi stringa di query).

    b. Per Nome immetti **`type`**.

    c. Mantieni Obbligatorio e Caching disattivati.

    Ripeti le fasi precedenti per creare una stringa di query aggiuntiva denominata **`page`**.

11. Scegli Crea metodo.

Se invii una richiesta, ora il client può fornire un tipo di animale e un numero di pagina come parametri della stringa di query. Questi parametri di input devono essere mappati nei parametri della stringa di query dell'integrazione per inoltrare i valori di input al nostro PetStore sito Web nel backend.

Per mappare i parametri di input alla richiesta di integrazione

1. Nella scheda Richiesta di integrazione scegli Modifica in Impostazioni della richiesta di integrazione.
2. Scegliete i parametri della stringa di URL query, quindi effettuate le seguenti operazioni:

    a. Scegli Aggiungi parametro della stringa di query.

    b. Per Nome, immetti **`type`**.

    c. Per Mappato da immetti **`method.request.querystring.type`**.

    d. Mantieni disattivata l'opzione Caching.

    e. Scegli Aggiungi parametro della stringa di query.

    f. Per Nome, immetti **`page`**.

    g. Per Mappato da immetti **`method.request.querystring.page`**.

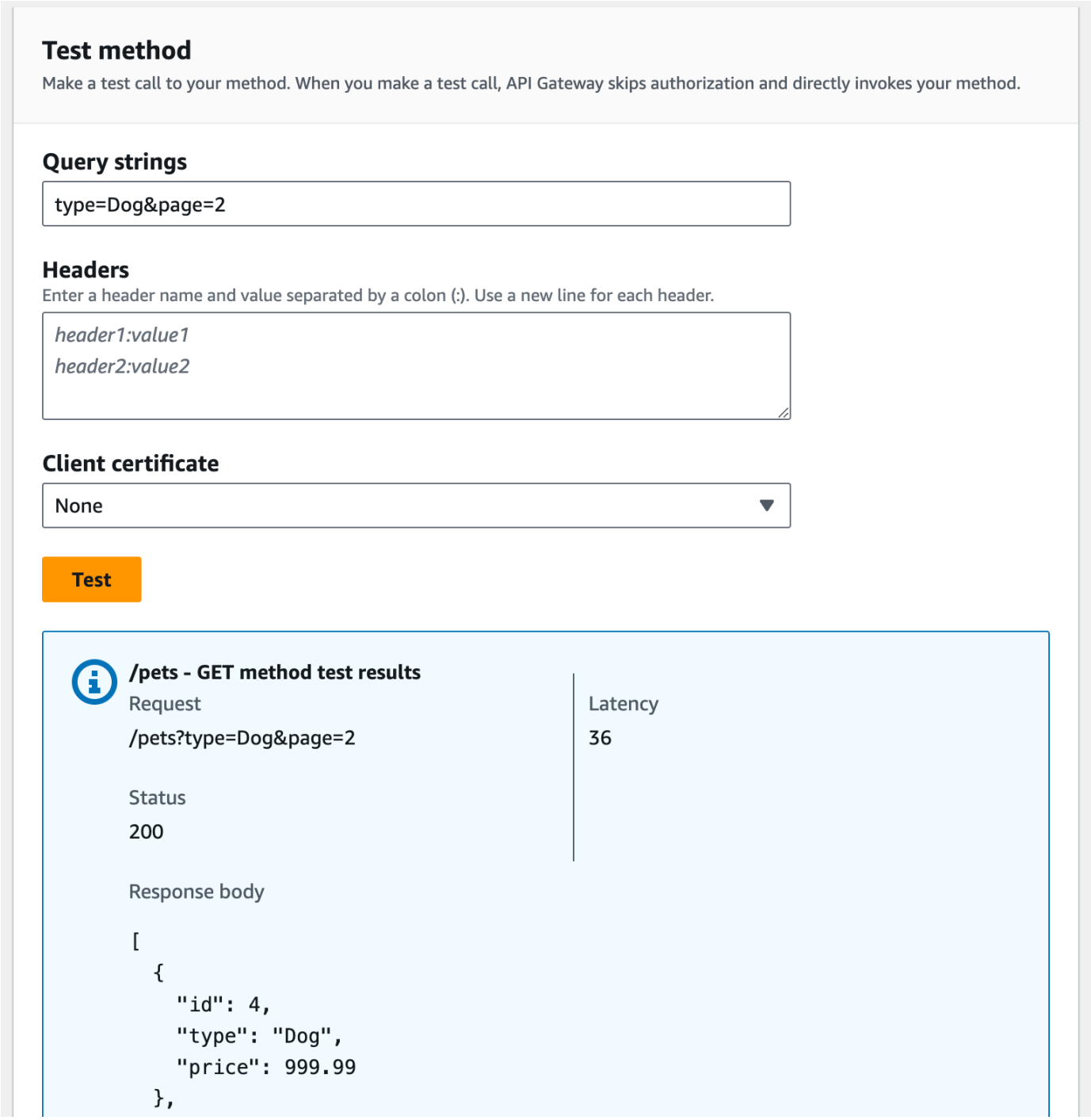
    h. Mantieni disattivata l'opzione Caching.

3. Selezionare Salva.

Per testare il API

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. In Stringhe di query immetti **`type=Dog&page=2`**.
3. Scegli Test (Esegui test).

   Il risultato è simile al seguente:



Ora che il test ha avuto esito positivo, possiamo implementarlo API per renderlo disponibile al pubblico.

4. Seleziona Deploy (Implementa)API.
5. In Fase, seleziona Nuova fase.
6. In Stage name (Nome fase) immettere **`Prod`**.

7. (Facoltativo) In Description (Descrizione), immettere una descrizione.
8. Seleziona Deploy (Implementa).
9. (Facoltativo) Nella sezione Dettagli dello stage, per Invoke URL, puoi scegliere l'icona di copia per copiare la tua API chiamata. URL Puoi usarla con strumenti come Postman e c URL per testare il tuo. API

Se usi an SDK per creare un client, puoi chiamare i metodi esposti da SDK per firmare la richiesta. Per i dettagli sull'implementazione, consulta quello AWS SDKdi tua scelta.

> **Note**
>
> Quando vengono apportate modifiche alla tuaAPI, devi ridistribuirla API per rendere disponibili le funzionalità nuove o aggiornate prima di richiamare nuovamente la richiesta. URL

(Facoltativo) Mappatura dei parametri di richiesta

Mappa i parametri di richiesta per un gateway API API

Questo tutorial mostra come creare un parametro path di `{petId}` on the API method request URL per specificare un ID elemento, mapparlo al parametro `{id}` path nella richiesta URL di integrazione e inviare la richiesta all'endpoint. HTTP

> **Note**
>
> Se immetti una lettera maiuscola o minuscola errata, ad esempio una lettera minuscola anziché maiuscola, verranno restituiti errori più avanti nella procedura dettagliata.

Fase 1: Creazione delle risorse

In questo passaggio, crei una risorsa con un parametro path {petId}.

Per creare la risorsa {petId}

1. Seleziona la risorsa /pets, quindi scegli Crea risorsa.
2. Mantieni l'opzione Risorsa proxy disattivata.
3. Per Percorso risorsa seleziona /pets.

4. Per Resource Name (Nome risorsa) immetti **`{petId}`**.

   Usa le parentesi graffe (`{ }`) in `petId` modo che /pets/ {} sia visualizzato. petId
5. Mantieni CORS(Cross Origin Resource Sharing) disattivata.
6. Scegli Crea risorsa.

Fase 2: Creazione e test dei metodi

In questa fase crei un metodo `GET` con un parametro di percorso `{petId}`.

Per impostare il GET metodo

1. Seleziona la risorsa/{petId}, quindi scegli il metodo Create.
2. In Tipo di metodo, seleziona GET.
3. Per Tipo di integrazione, seleziona HTTPintegrazione.
4. Mantieni disattivata l'integrazione del HTTP proxy.
5. Per HTTPmetodo, seleziona GET.
6. Per EndpointURL, immettere **`http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}`**
7. Per Gestione contenuti seleziona Transito.
8. Mantieni attivata l'opzione Timeout predefinito.
9. Scegli Crea metodo.

Ora mappate il parametro `{petId}` path al parametro `{id}` path nell'HTTPendpoint.

Per mappare il parametro di percorso **`{petId}`**

1. Nella scheda Richiesta di integrazione scegli Modifica in Impostazioni della richiesta di integrazione.
2. Scegliete i parametri URL del percorso.
3. APIGateway crea un parametro di percorso per la richiesta di integrazione denominata petId. Questo non funziona per il tuo backend. L'HTTPendpoint utilizza `{id}` come parametro del percorso. Rinomina petId come **`id`**.

   In questo modo, verrà mappato il parametro di percorso `petId` della richiesta del metodo al parametro di percorso `id` della richiesta di integrazione.

4. Selezionare Salva.

Quindi testa il metodo.

Test del metodo

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. In Percorso per petId, immettete**4**.
3. Scegli Test (Esegui test).

   Se il test ha esito positivo, in Corpo della risposta viene visualizzato quanto segue:

   ```
   {
     "id": 4,
     "type": "bird",
     "price": 999.99
   }
   ```

Fase 3: Implementare il API

In questo passaggio, si distribuisce il in API modo da poter iniziare a chiamarlo al di fuori della console API Gateway.

Per distribuire API

1. Seleziona Deploy (Implementa)API.
2. Per Fase seleziona Produzione.
3. (Facoltativo) In Description (Descrizione), immettere una descrizione.
4. Seleziona Deploy (Implementa).

Fase 4: Test del API

In questo passaggio, esci dalla console API Gateway e utilizzi la tua API per accedere all'HTTPendpoint.

1. Nel pannello di navigazione principale scegli Fase.
2. Nella sezione Dettagli dello stage, scegli l'icona di copia per copiare API la tua chiamataURL.

Dovrebbe essere simile a quanto segue:

```
https://my-api-id.execute-api.region-id.amazonaws.com/prod
```

3. URLInseriscila nella casella dell'indirizzo di una nuova scheda del browser e aggiungila URL prima di `/pets/4` inviare la richiesta.
4. Il browser restituirà quanto segue:

```
{
  "id": 4,
  "type": "bird",
  "price": 999.99
}
```

Passaggi successivi

Puoi personalizzare ulteriormente la tua API attivando la convalida della richiesta, trasformando i dati o creando risposte gateway personalizzate.

Per scoprire altri modi per personalizzare il tuoAPI, consulta i seguenti tutorial:

- Per ulteriori informazioni sulla convalida della richiesta, consulta Imposta la convalida delle richieste di base in API Gateway.
- Per informazioni su come trasformare i payload di richiesta e risposta, consulta Configura le trasformazioni dei dati in API Gateway.
- Per informazioni su come creare risposte del gateway personalizzate, consulta Imposta una risposta gateway per un REST API utilizzo della console Gateway API.

Tutorial: crea un'API REST con un'integrazione privata

Puoi creare un'API di API Gateway con integrazione privata per fornire ai clienti l'accesso alle risorse HTTP/HTTPS in Amazon Virtual Private Cloud (Amazon VPC). Tali risorse VPC sono endpoint HTTP/HTTPS su un'istanza EC2 regolati da un Network Load Balancer nel VPC. Network Load Balancer incapsula la risorsa VPC e instrada le richieste in ingresso alla risorsa di destinazione.

Quando un client chiama l'API, API Gateway si connette a Network Load Balancer mediante il collegamento VPC preconfigurato. Un collegamento VPC è incapsulato da una risorsa API Gateway

di. VpcLink È responsabile dell'inoltro delle richieste del metodo API alle risorse VPC e restituisce le risposte di back-end all'intermediario. Per uno sviluppatore di API, `VpcLink` è funzionalmente equivalente a un endpoint di integrazione.

Per creare un'API con integrazione privata, devi crearne una nuova o scegliere un `VpcLink` esistente connesso a un Network Load Balancer che si rivolge alle risorse VPC desiderate. È necessario disporre delle autorizzazioni appropriate per creare e gestire un `VpcLink`. Configura quindi un metodo API e integralo con il `VpcLink` impostando `HTTP` o `HTTP_PROXY` come il tipo di integrazione, `VPC_LINK` come il tipo di connessione dell'integrazione e l'identificativo `VpcLink` per il `connectionId` dell'integrazione.

> **Note**
>
> Il Network Load Balancer e l'API devono appartenere allo stesso AWS account.

Per iniziare subito a creare un'API per accedere alle risorse VPC, illustreremo le fasi essenziali per la creazione di un'API con l'integrazione privata utilizzando la console API Gateway. Prima di creare l'API, esegui le operazioni seguenti:

1. Crea una risorsa VPC, crea o scegli un Network Load Balancer con il tuo account nella stessa regione e aggiungi l'istanza EC2 che ospita la risorsa come destinazione del Network Load Balancer. Per ulteriori informazioni, consulta Configurare un Network Load Balancer per le integrazioni private del API gateway.
2. Concedi le autorizzazioni per creare i collegamenti VPC per le integrazioni private. Per ulteriori informazioni, consulta Concedi a API Gateway le autorizzazioni per creare un collegamento VPC.

Dopo la creazione della risorsa VPC e del Network Load Balancer con la risorsa VPC configurata nei gruppi di destinazione, segui le istruzioni riportate di seguito per creare un'API e integrala con la risorsa VPC mediante un `VpcLink` in un'integrazione privata.

Per creare un'API con un'integrazione privata

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Se si utilizza API Gateway per la prima volta, verrà visualizzata una pagina che presenta le caratteristiche del servizio. In API REST, scegliere Crea. Quando appare il popup Create Example API (Crea API di esempio), scegliere OK.

Se non è la prima volta che si utilizza API Gateway, scegliere Create API (Crea API). In API REST, scegliere Crea.

3. Crea una REST API regionale o ottimizzata per edge.
4. Seleziona l'API.
5. Scegli Crea metodo e procedi come descritto di seguito:

   a. In Tipo di metodo, seleziona `GET`.
   b. In Tipo di integrazione seleziona Collegamento VPC.
   c. Attiva l'opzione Integrazione proxy VPC.
   d. Per Metodo HTTP seleziona `GET`.
   e. Per Collegamento VPC seleziona [Utilizza le varabili di fase] e immetti **`${stageVariables.vpcLinkId}`** nella casella di testo sottostante.

      Definisci la variabile di fase `vpcLinkId` dopo l'implementazione dell'API in una fase e imposti il suo valore sull'ID del `VpcLink`.
   f. Immetti un URL, ad esempio `http://myApi.example.com`, per URL dell'endpoint.

      Qui viene utilizzato il nome host (ad esempio, `myApi.example.com`) per impostare l'intestazione `Host` della richiesta di integrazione.
   g. Scegli Crea metodo.

      Con l'integrazione proxy, l'API è pronta per la distribuzione. In caso contrario, è necessario procedere con la configurazione delle risposte di metodo e delle risposte di integrazione appropriate.
6. Scegli Distribuisci l'API e procedi come indicato di seguito:

   a. In Fase, seleziona Nuova fase.
   b. Per Nome fase immetti il nome di una fase.
   c. (Facoltativo) In Description (Descrizione), immettere una descrizione.
   d. Seleziona Deploy (Implementa).
7. Nella sezione Dettagli fase annota l'URL di richiamata risultante. Ti servirà per invocare l'API. Prima di farlo, dovrai configurare la variabile di fase `vpcLinkId`.
8. Nel riquadro Fasi scegli la scheda Variabili di fase, quindi procedi come indicato di seguito:

   a. Scegli Gestisci variabili, quindi seleziona Aggiungi variabile di fase.

b. Per Nome, immetti **`vpcLinkId`**.

c. Per Valore immetti l'ID del `VPC_LINK`, ad esempio *`gix6s7`*.

d. Selezionare Salva.

Se utilizzi la variabile di fase, modificandone il valore puoi facilmente passare a collegamenti VPC diversi per l'API.

Tutorial: creare un'API REST con un' AWS integrazione

Gli argomenti Tutorial: Creare un'integrazione REST API con un proxy Lambda e Tutorial: Creare un'integrazione non proxy REST API con Lambda descrivono entrambi come creare un'API di API Gateway per esporre la funzione Lambda integrata. Inoltre, puoi creare un'API API Gateway per esporre altri AWS servizi, come Amazon SNS, Amazon S3, Amazon Kinesis e persino. AWS Lambda Questo è possibile grazie all'integrazione di `AWS`. L'integrazione Lambda o l'integrazione proxy Lambda è un caso speciale, in cui la chiamata della funzione Lambda viene esposta tramite l'API di API Gateway.

Tutti i AWS servizi supportano API dedicate per esporre le proprie funzionalità. Tuttavia, è probabile che i protocolli applicativi o le interfacce di programmazione differiscano da servizio a servizio. Un'API API Gateway con `AWS` integrazione ha il vantaggio di fornire un protocollo applicativo coerente per consentire al cliente di accedere a diversi AWS servizi.

In questa procedura guidata creeremo un'API per esporre Amazon SNS. Per altri esempi di integrazione di un'API con altri AWS servizi, consultaTutorial e workshop di Amazon API Gateway.

Diversamente dall'integrazione proxy Lambda, non esiste un'integrazione proxy corrispondente per altri servizi AWS . Pertanto, un metodo API è integrato con una singola AWS azione. Per ottenere maggiore flessibilità, analogamente all'integrazione proxy, puoi configurare l'integrazione proxy Lambda. La funzione Lambda quindi analizza ed elabora le richieste di altre azioni. AWS

API Gateway non esegue un nuovo tentativo quando si verifica il timeout dell'endpoint. La chiamata API deve implementare una logica di ripetizione per gestire i timeout degli endpoint.

Questa procedura dettagliata si basa sulle istruzioni e sui concetti riportati in Tutorial: Creare un'integrazione non proxy REST API con Lambda. Se la procedura dettagliata non è ancora stata completata, è consigliabile eseguirla ora prima di proseguire.

Argomenti

- Prerequisiti
- Fase 1: creare il ruolo di esecuzione del proxy del AWS servizio
- Fase 2: creazione della risorsa
- Fase 3: creazione del metodo GET
- Fase 4: specifica delle impostazioni del metodo e test del metodo
- Fase 5: distribuzione dell'API
- Fase 6: test dell'API
- Fase 7: pulire

Prerequisiti

Prima di iniziare questa procedura guidata, esegui queste operazioni:

1. Completa le fasi descritte in Configurazione per l'utilizzo di API Gateway.
2. Crea una nuova API denominata `MyDemoAPI`. Per ulteriori informazioni, consulta Tutorial: creare un file REST API con un'HTTPintegrazione non proxy.
3. Distribuisci l'API almeno una volta in una fase denominata `test`. Per ulteriori informazioni, consulta la pagina relativa alla distribuzione dell'API in Scegli un tutorial di integrazione AWS Lambda.
4. Completa le altre fasi descritte in Scegli un tutorial di integrazione AWS Lambda.
5. Crea almeno un argomento in Amazon Simple Notification Service (Amazon SNS). Utilizzerai l'API distribuita per ottenere un elenco di argomenti in Amazon SNS associati AWS al tuo account. Per informazioni su come creare un argomento in Amazon SNS, consulta Creazione di un argomento. Non devi copiare l'ARN dell'argomento citato nella fase 5.

Fase 1: creare il ruolo di esecuzione del proxy del AWS servizio

Per permettere all'API di richiamare le operazioni Amazon SNS necessarie, devi aver collegato le policy IAM appropriate a un ruolo IAM.

Per creare il ruolo di esecuzione del proxy di AWS servizio

1. Accedi AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.

2. Scegliere Roles (Ruoli).
3. Scegli Crea ruolo.
4. Scegli il AWS servizio in Seleziona il tipo di entità affidabile, quindi seleziona API Gateway e seleziona Consenti a API Gateway di inviare i log ai CloudWatch log.
5. Scegli Successivo e di nuovo Successivo.
6. In Role name (Nome ruolo) immettere **`APIGatewaySNSProxyPolicy`** e quindi selezionare Create role (Crea ruolo).
7. Nell'elenco Roles (Ruoli) scegliere il ruolo appena creato. Potrebbe essere necessario scorrere o utilizzare la barra di ricerca per trovare il ruolo.
8. Per il ruolo selezionato, seleziona la scheda Aggiungi autorizzazioni.
9. Dall'elenco a discesa scegli Collega policy.
10. Nella barra di ricerca inserisci **`AmazonSNSReadOnlyAccess`** e scegli Aggiungi autorizzazioni.

> **Note**
>
> Per semplicità, questo tutorial utilizza una policy gestita. Come best practice, dovresti creare la tua policy IAM per concedere le autorizzazioni minime richieste.

11. Annota l'ARN del ruolo appena creato, lo utilizzerai in seguito.

Fase 2: creazione della risorsa

In questo passaggio, crei una risorsa che consente al proxy del AWS servizio di interagire con il AWS servizio.

Per creare la risorsa

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere l'API.
3. Seleziona la risorsa root /, rappresentata da una singola barra (/), quindi scegli Crea risorsa.
4. Mantieni l'opzione Risorsa proxy disattivata.
5. Mantieni Percorso risorsa impostato su `/`.
6. Per Resource Name (Nome risorsa) immetti **`mydemoawsproxy`**.
7. Mantieni CORS (Cross Origin Resource Sharing) disattivato.

8. Scegli Crea risorsa.

Fase 3: creazione del metodo GET

In questo passaggio, si crea un metodo GET che consente al proxy del AWS servizio di interagire con il AWS servizio.

Per creare il metodo **`GET`**

1. Seleziona la risorsa /mydemoawsproxy, quindi scegli Crea metodo.
2. Per Tipo di metodo seleziona GET.
3. Per Tipo di integrazione seleziona Servizio AWS.
4. Per Regione AWS, seleziona l'argomento Regione AWS in cui hai creato il tuo Amazon SNS.
5. Per Servizio AWS seleziona Amazon SNS.
6. Lascia vuoto Sottodominio AWS .
7. Per Metodo HTTP seleziona GET.
8. Per Tipo di operazione scegli Usa nome operazione.
9. Per Nome azione immetti **`ListTopics`**.
10. Per Ruolo di esecuzione immetti l'ARN del ruolo per **`APIGatewaySNSProxyPolicy`**.
11. Scegli Crea metodo.

Fase 4: specifica delle impostazioni del metodo e test del metodo

A questo punto puoi testare il metodo `GET` per verificare che sia stato configurato correttamente per elencare gli argomenti Amazon SNS.

Test del metodo **`GET`**

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. Scegli Test (Esegui test).

   Il risultato mostra una risposta simile a quella riportata di seguito:

```
{
```

```
  "ListTopicsResponse": {
    "ListTopicsResult": {
      "NextToken": null,
      "Topics": [
        {
          "TopicArn": "arn:aws:sns:us-east-1:80398EXAMPLE:MySNSTopic-1"
        },
        {
          "TopicArn": "arn:aws:sns:us-east-1:80398EXAMPLE:MySNSTopic-2"
        },
        ...
        {
          "TopicArn": "arn:aws:sns:us-east-1:80398EXAMPLE:MySNSTopic-N"
        }
      ]
    },
    "ResponseMetadata": {
      "RequestId": "abc1de23-45fa-6789-b0c1-d2e345fa6b78"
    }
  }
}
```

Fase 5: distribuzione dell'API

In questa fase, viene distribuita l'API per poterla chiamare esternamente alla console API Gateway.

Per distribuire l'API

1. Seleziona Deploy API (Distribuisci API).
2. In Fase, seleziona Nuova fase.
3. In Stage name (Nome fase) immettere **`test`**.
4. (Facoltativo) In Description (Descrizione), immettere una descrizione.
5. Selezionare Deploy (Distribuisci).

Fase 6: test dell'API

In questa fase, esci dalla console API Gateway e utilizzi il proxy del AWS servizio per interagire con il servizio Amazon SNS.

1. Nel pannello di navigazione principale scegli Fase.

2. In Dettagli fase, scegli l'icona Copia per copiare l'URL di richiamo dell'API.

   L'URL dovrebbe essere simile a questo:

```
https://my-api-id.execute-api.region-id.amazonaws.com/test
```

3. Immetti l'URL nella barra degli indirizzi di una nuova scheda del browser.
4. Aggiungi `/mydemoawsproxy` in modo che l'URL sia simile al seguente:

```
https://my-api-id.execute-api.region-id.amazonaws.com/test/mydemoawsproxy
```

   Accedi all'URL. Verranno visualizzate informazioni simili alle seguenti:

```
{"ListTopicsResponse":{"ListTopicsResult":{"NextToken": null,"Topics":
[{"TopicArn": "arn:aws:sns:us-east-1:80398EXAMPLE:MySNSTopic-1"},{"TopicArn":
 "arn:aws:sns:us-east-1:80398EXAMPLE:MySNSTopic-2"},...{"TopicArn":
 "arn:aws:sns:us-east-1:80398EXAMPLE:MySNSTopic-N}]},"ResponseMetadata":
{"RequestId":"abc1de23-45fa-6789-b0c1-d2e345fa6b78}}}
```

Fase 7: pulire

Puoi eliminare le risorse IAM necessarie al proxy del AWS servizio per funzionare.

> **Warning**
>
> Se elimini una risorsa IAM su cui si basa un proxy di AWS servizio, quel proxy di AWS servizio e tutte le API che si basano su di esso non funzioneranno più. L'eliminazione di una risorsa IAM non può essere annullata. Per usare di nuovo la risorsa IAM, è necessario ricrearla.

Per eliminare le risorse IAM associate

1. Aprire la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nell'area Details (Dettagli) scegliere Roles (Ruoli).
3. Seleziona APIGateway AWSProxyExecRole, quindi scegli Role Actions, Delete Role. Quando viene richiesto, scegliere Yes, Delete (Sì, elimina).
4. Nell'area Details (Dettagli) scegliere Policies (Policy).

5. Seleziona ApiGateway AWSProxyExecPolicy, quindi scegli Azioni politiche, Elimina. Quando viene richiesto, scegliere Delete (Elimina).

Hai completato questa procedura guidata. Per discussioni più dettagliate sulla creazione di API come proxy di AWS servizio, consultaTutorial: creare un REST API proxy come Amazon S3, Tutorial: crea una calcolatrice REST API con due integrazioni di AWS servizi e un'integrazione non proxy Lambda o. Tutorial: creazione di un'API REST come un proxy Amazon Kinesis

Tutorial: crea una calcolatrice REST API con due integrazioni di AWS servizi e un'integrazione non proxy Lambda

the section called “Tutorial: Creare un'integrazione non proxy REST API con Lambda”Utilizza esclusivamente `Lambda Function` l'integrazione. `Lambda Function`l'integrazione è un caso speciale del tipo di `AWS Service` integrazione che esegue al posto tuo gran parte della configurazione dell'integrazione, ad esempio l'aggiunta automatica delle autorizzazioni basate sulle risorse richieste per richiamare la funzione Lambda. In questo caso, due integrazioni su tre utilizzano l'integrazione `AWS Service`. In questo tipo di integrazione, hai un maggiore controllo, ma dovrai eseguire manualmente attività come creare e specificare un ruolo contenente le autorizzazioni appropriate. IAM

In questo tutorial, creerai una funzione `Calc` Lambda che implementa operazioni aritmetiche di base, accettando e restituendo input e output formattati. JSON Quindi creerai una funzione REST API e la integrerai con la funzione Lambda nei seguenti modi:

1. Esponendo un metodo `GET` nella risorsa `/calc` per richiamare la funzione Lambda, fornendo l'input sotto forma di parametri delle stringhe di query (integrazione `AWS Service`).
2. Esponendo un metodo `POST` nella risorsa `/calc` per richiamare la funzione Lambda, fornendo l'input nel payload della richiesta del metodo (integrazione `AWS Service`).
3. Esponendo un metodo `GET` nelle risorse `/calc/{operand1}/{operand2}/{operator}` nidificate per richiamare la funzione Lambda, fornendo l'input sotto forma di parametri di percorso (integrazione `Lambda Function`).

Oltre a provare questo tutorial, potresti voler studiare il file di API definizione aperto per `Calc`API, che puoi importare in API Gateway seguendo le istruzioni riportate inthe section called “Apri API”.

Argomenti

- Creazione di un ruolo IAM assumibile
- Creazione di una funzione Lambda Calc
- Test della funzione Lambda Calc
- Crea un Calc API
- Integrazione 1: creazione di un metodo GET con parametri di query per chiamare la funzione Lambda
- Integrazione 2: creazione di un POST metodo con un JSON payload per chiamare la funzione Lambda
- Integrazione 3: creazione di un metodo GET con parametri di percorso per chiamare la funzione Lambda
- APIDefinizioni aperte del campione API integrate con una funzione Lambda

Creazione di un ruolo IAM assumibile

Per poter API richiamare la funzione `Calc` Lambda, è necessario disporre di un ruolo Gateway IAM assumibile, ovvero API IAM un ruolo con la seguente relazione di fiducia:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "apigateway.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Il ruolo che crei dovrà avere l'autorizzazione Lambda InvokeFunction. In caso contrario, il API chiamante riceverà una `500 Internal Server Error` risposta. Per concedere al ruolo questa autorizzazione, allegherai la seguente IAM politica:

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": "*"
        }
    ]
}
```

Ecco come procedere:

Crea un ruolo assumibile in API IAM Gateway

1. Accedi alla console di IAM.
2. Scegliere Roles (Ruoli).
3. Seleziona Create Role (Crea ruolo).
4. In Select type of trusted entity (Seleziona tipo di entità attendibile), scegli AWS Service.
5. In Choose the service that will use this role (Scegli il servizio che utilizzerà questo ruolo), seleziona Lambda.
6. Scegli Next: Permissions (Successivo: Autorizzazioni).
7. Selezionare Create Policy (Crea policy).

   Si aprirà una nuova finestra Create Policy (Crea policy) della console. Nella finestra procedi nel seguente modo:

   a. Nella JSONscheda, sostituisci la politica esistente con la seguente:

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": "lambda:InvokeFunction",
               "Resource": "*"
           }
       ]
   }
   ```

   b. Scegliere Review policy (Esamina policy).

c. InReview Policy (Rivedi policy), effettua le operazioni seguenti:

   i. In Name (Nome), digita un nome, ad esempio **`lambda_execute`**.

   ii. Selezionare Create Policy (Crea policy).

8. Nella finestra Create Role (Crea ruolo) originaria della console, procedi nel seguente modo:

   a. In Attach permissions policies (Collega policy delle autorizzazioni), scegli la tua policy **`lambda_execute`** dall'elenco a discesa.

      Se la policy non è visualizzata nell'elenco, scegliere il pulsante di aggiornamento in alto. Non aggiornare la pagina del browser.

   b. Scegliere Next: Tags (Successivo: Tag).

   c. Scegli Next:Review (Successivo:Rivedi).

   d. Per Role name (Nome ruolo), digita un nome, ad esempio **`lambda_invoke_function_assume_apigw_role`**.

   e. Seleziona Create role (Crea ruolo).

9. Seleziona il tuo **`lambda_invoke_function_assume_apigw_role`** nell'elenco dei ruoli.

10. Seleziona la scheda Trust relationships (Relazioni di trust).

11. Seleziona Edit trust relationship (Modifica relazione di trust).

12. Sostituisci la policy esistente con la seguente:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "lambda.amazonaws.com",
          "apigateway.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

13. Selezionare Update Trust Policy (Aggiorna policy di trust).
14. Prendi nota del ruolo ARN per il ruolo che hai appena creato. Sarà necessario in seguito.

Creazione di una funzione Lambda **`Calc`**

Verrà quindi creata una funzione Lambda utilizzando la console Lambda.

1. Nella console Lambda, scegliere Create function (Crea funzione).
2. Scegli Author from Scratch (Crea da zero).
3. In Name (Nome), immettere **`Calc`**.
4. Per Runtime, scegli l'ultimo runtime supportato di Node.js o di Python.
5. Per tutte le altre opzioni, usa l'impostazione predefinita.
6. Scegli Crea funzione.
7. Copia la seguente funzione Lambda e incollala nel runtime preferito e nell'editor di codice nella console Lambda.

   Node.js

```
export const handler = async function (event, context) {
  console.log("Received event:", JSON.stringify(event));

  if (
    event.a === undefined ||
    event.b === undefined ||
    event.op === undefined
  ) {
    return "400 Invalid Input";
  }

  const res = {};
  res.a = Number(event.a);
  res.b = Number(event.b);
  res.op = event.op;
  if (isNaN(event.a) || isNaN(event.b)) {
    return "400 Invalid Operand";
  }
  switch (event.op) {
    case "+":
```

```
    case "add":
      res.c = res.a + res.b;
      break;
    case "-":
    case "sub":
      res.c = res.a - res.b;
      break;
    case "*":
    case "mul":
      res.c = res.a * res.b;
      break;
    case "/":
    case "div":
      if (res.b == 0) {
        return "400 Divide by Zero";
      } else {
        res.c = res.a / res.b;
      }
      break;
    default:
      return "400 Invalid Operator";
  }

  return res;
};
```

Python

```
import json


def lambda_handler(event, context):
    print(event)

    try:
        (event['a']) and (event['b']) and (event['op'])
    except KeyError:
        return '400 Invalid Input'

    try:
        res = {
            "a": float(
                event['a']), "b": float(
```

```
                event['b']), "op": event['op']}
    except ValueError:
        return '400 Invalid Operand'

    if event['op'] == '+':
        res['c'] = res['a'] + res['b']
    elif event['op'] == '-':
        res['c'] = res['a'] - res['b']
    elif event['op'] == '*':
        res['c'] = res['a'] * res['b']
    elif event['op'] == '/':
        if res['b'] == 0:
            return '400 Divide by Zero'
        else:
            res['c'] = res['a'] / res['b']
    else:
        return '400 Invalid Operator'

    return res
```

8. In Execution role (Ruolo di esecuzione) scegliere Choose an existing role (Scegli un ruolo esistente).
9. Inserisci il ruolo ARN per il **`lambda_invoke_function_assume_apigw_role`** ruolo che hai creato in precedenza.
10. Selezionare Deploy (Distribuisci).

Questa funzione richiede due operandi (`a` e `b`) e un operatore (`op`) del parametro di input `event`. L'input è un JSON oggetto del seguente formato:

```
{
  "a": "Number" | "String",
  "b": "Number" | "String",
  "op": "String"
}
```

Questa funzione restituisce il risultato calcolato (`c`) e l'input. Per un input non valido, la funzione restituisce il valore null o la stringa "Invalid op" come risultato. L'output ha il seguente JSON formato:

```
{
  "a": "Number",
  "b": "Number",
  "op": "String",
  "c": "Number" | "String"
}
```

Dovresti testare la funzione nella console Lambda prima di integrarla con la API nel passaggio successivo.

Test della funzione Lambda **`Calc`**

Di seguito viene descritto come testare la funzione `Calc` nella console Lambda:

1. Seleziona la scheda Test.
2. Come nome dell'evento di test, immettere **`calc2plus5`**.
3. Sostituire la definizione dell'evento di test con quanto segue:

   ```
   {
     "a": "2",
     "b": "5",
     "op": "+"
   }
   ```

4. Scegli Save (Salva).
5. Scegli Test (Esegui test).
6. Espandere Execution result: succeeded (Risultato esecuzione: riuscita). Verrà visualizzato un codice analogo al seguente:

   ```
   {
     "a": 2,
     "b": 5,
     "op": "+",
     "c": 7
   }
   ```

Crea un **`Calc`** API

La procedura seguente mostra come creare una API per la funzione `Calc` Lambda appena creata. Nelle sezioni successive verranno aggiunti metodi e risorse.

Per creare un API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Se è la prima volta che utilizzi API Gateway, viene visualizzata una pagina che presenta le funzionalità del servizio. In RESTAPI, scegli Costruisci. Quando viene visualizzato il API popup Crea esempio, scegli OK.

   Se non è la prima volta che usi API Gateway, scegli Crea API. In RESTAPI, scegli Build.
3. Per APInome, inserisci**`LambdaCalc`**.
4. (Facoltativo) In Description (Descrizione), immettere una descrizione.
5. Mantieni il tipo di API endpoint impostato su Regionale.
6. Scegli Crea API.

Integrazione 1: creazione di un metodo **`GET`** con parametri di query per chiamare la funzione Lambda

Creando un `GET` metodo che passa i parametri della stringa di query alla funzione Lambda, si abilita la API loro invocazione da un browser. Questo approccio può essere utile, soprattutto perché consente APIs l'accesso aperto.

Dopo aver creato unAPI, crei una risorsa. In genere, API le risorse sono organizzate in un albero di risorse in base alla logica dell'applicazione. In questa fase crei una risorsa /calc.

Per creare una risorsa /calc

1. Scegli Crea risorsa.
2. Mantieni l'opzione Risorsa proxy disattivata.
3. Mantieni Percorso risorsa impostato su `/`.
4. Per Resource Name (Nome risorsa) immetti **`calc`**.
5. Mantieni CORS(Cross Origin Resource Sharing) disattivato.
6. Scegli Crea risorsa.

Creando un `GET` metodo che passa i parametri della stringa di query alla funzione Lambda, si abilita la API loro invocazione da un browser. Questo approccio può essere utile, soprattutto perché consente APIs l'accesso aperto.

Con questo metodo, Lambda richiede che venga usata la richiesta `POST` per richiamare qualsiasi funzione Lambda. Questo esempio mostra che il HTTP metodo in una richiesta di metodo frontend può essere diverso dalla richiesta di integrazione nel backend.

Creazione di un metodo **`GET`**

1. Seleziona la risorsa /calc quindi scegli Crea metodo.
2. In Tipo di metodo, seleziona GET.
3. Per Tipo di integrazione seleziona Servizio AWS.
4. Per Regione AWS, seleziona il Regione AWS luogo in cui hai creato la tua funzione Lambda.
5. Per Servizio AWS seleziona Lambda.
6. Lascia vuoto Sottodominio AWS .
7. Per HTTPmetodo, seleziona POST.
8. In Tipo di operazione scegli Utilizza sostituzione percorso. Questa opzione ci consente ARN di specificare l'azione Invoke per eseguire la nostra `Calc` funzione.
9. Per Sostituzione percorso immetti **`2015-03-31/functions/arn:aws:lambda:`*`us-east-2`*`:`*`account-id`*`:function:Calc/invocations`**. Per *`account-id`*, inserisci il tuo Account AWS ID. Per*`us-east-2`*, inserisci Regione AWS dove hai creato la tua funzione Lambda.
10. Per Ruolo di esecuzione, inserisci il ruolo ARN per**`lambda_invoke_function_assume_apigw_role`**.
11. Non modificare le impostazioni di Cache delle credenziali e Timeout predefinito.
12. Scegliete le impostazioni di richiesta del metodo.
13. Per Validatore richiesta seleziona Convalida parametri di stringa query e intestazioni.

    Questa impostazione restituisce un messaggio di errore se il client non specifica i parametri obbligatori.
14. Scegli i parametri URL della stringa di query.

    Ora impostate i parametri della stringa di query per il GETmetodo sulla risorsa /calc in modo che possa ricevere input per conto della funzione Lambda di backend.

Per creare i parametri della stringa di query, procedi come segue:

a. Scegliere Add query string (Aggiungi stringa di query).
b. Per Nome, immetti **`operand1`**.
c. Attiva Campo obbligatorio.
d. Mantieni disattivata l'opzione Caching.

Ripeti le stesse fasi e crea una stringa di query denominata **`operand2`** e una stringa di query denominata **`operator`**.

15. Scegli Crea metodo.

A questo punto crea un modello di mappatura per tradurre le stringhe di query fornite dal client in payload di richiesta di integrazione, come richiesto dalla funzione `Calc`. Questo modello mappa i tre parametri della stringa di query dichiarati nella richiesta del metodo nei valori delle proprietà designati dell'JSONoggetto come input per la funzione Lambda di backend. L'JSONoggetto trasformato verrà incluso come payload della richiesta di integrazione.

Per mappare i parametri di input alla richiesta di integrazione

1. Nella scheda Richiesta di integrazione scegli Modifica in Impostazioni della richiesta di integrazione.
2. Per Richiesta corpo passthrough scegli Quando non ci sono modelli definiti (consigliato).
3. Scegli Modelli di mappatura.
4. Scegliere Add mapping template (Aggiungi modello di mappatura).
5. Per Tipo di contenuto inserisci **`application/json`**.
6. Per Corpo del modello inserisci il seguente codice:

```
{
    "a":  "$input.params('operand1')",
    "b":  "$input.params('operand2')",
    "op": "$input.params('operator')"
}
```

7. Seleziona Salva.

È ora possibile testare il metodo `GET` per verificare che sia stato configurato correttamente per richiamare la funzione Lambda.

Test del metodo **`GET`**

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. In Stringhe di query immetti **`operand1=2&operand2=3&operator=+`**.
3. Scegli Test (Esegui test).

   I risultati dovrebbero essere simili a quanto segue:

Test method

Make a test call to your method. When you make a test call, API Gateway skips authorization and directly invokes your method.

Query strings

operand1=2&operand2=3&operator=+

Headers

Enter a header name and value separated by a colon (:). Use a new line for each header.

header1:value1
header2:value2

Client certificate

None

Test

/ - GET method test results

Request

/?
operand1=2&operand2=3&operator=+

Status

200

Latency

414

Response body

```
{"a":2,"b":3,"op":"+","c":5}
```

Integrazione 2: creazione di un **POST** metodo con un JSON payload per chiamare la funzione Lambda

Creando un `POST` metodo con un JSON payload per chiamare la funzione Lambda, si fa in modo che il client debba fornire l'input necessario alla funzione di backend nel corpo della richiesta. Per garantire che il client carichi i dati di input corretti, verrà abilitata la convalida delle richieste nel payload.

Per creare un **POST** metodo con un payload JSON

1. Seleziona la risorsa /calc quindi scegli Crea metodo.
2. In Tipo di metodo, seleziona POST.
3. Per Tipo di integrazione seleziona Servizio AWS.
4. Per Regione AWS, seleziona il Regione AWS luogo in cui hai creato la tua funzione Lambda.
5. Per Servizio AWS seleziona Lambda.
6. Lascia vuoto Sottodominio AWS .
7. Per HTTPmetodo, seleziona POST.
8. In Tipo di operazione scegli Utilizza sostituzione percorso. Questa opzione ci consente ARN di specificare l'azione Invoke per eseguire la nostra `Calc` funzione.
9. Per Sostituzione percorso immetti **`2015-03-31/functions/arn:aws:lambda:`*`us-east-2`*`:`*`account-id`*`:function:Calc/invocations`**. Per ***`account-id`***, inserisci il tuo Account AWS ID. Per***`us-east-2`***, inserisci Regione AWS dove hai creato la tua funzione Lambda.
10. Per Ruolo di esecuzione, inserisci il ruolo ARN per**`lambda_invoke_function_assume_apigw_role`**.
11. Non modificare le impostazioni di Cache delle credenziali e Timeout predefinito.
12. Scegli Crea metodo.

A questo punto crea un modello di input per descrivere la struttura dei dati di input e convalidare il corpo della richiesta in ingresso.

Per creare un modello di input

1. Nel riquadro di navigazione principale seleziona Modelli.
2. Scegli Crea modello.

3. Per Nome, immetti **`input`**.
4. Per Tipo di contenuto inserisci **`application/json`**.

   Se non viene trovato alcun tipo di contenuto corrispondente, la convalida della richiesta non viene eseguita. Per utilizzare lo stesso modello indipendentemente dal tipo di contenuti, inserisci **`$default`**.
5. Per Schema modello immetti il seguente modello:

```
{
    "type":"object",
    "properties":{
        "a":{"type":"number"},
        "b":{"type":"number"},
        "op":{"type":"string"}
    },
    "title":"input"
}
```

6. Scegli Crea modello.

A questo punto crea un modello di output. Questo modello descrive la struttura di dati dell'output calcolato del back-end. Può essere usato per mappare i dati della risposta di integrazione a un modello diverso. Questo tutorial è basato sul comportamento di passthrough e non usa questo modello.

Per creare un modello di output

1. Scegli Crea modello.
2. Per Nome, immetti **`output`**.
3. Per Tipo di contenuto inserisci **`application/json`**.

   Se non viene trovato alcun tipo di contenuto corrispondente, la convalida della richiesta non viene eseguita. Per utilizzare lo stesso modello indipendentemente dal tipo di contenuti, inserisci **`$default`**.
4. Per Schema modello immetti il seguente modello:

```
{
    "type":"object",
    "properties":{
```

```
        "c":{"type":"number"}
    },
    "title":"output"
}
```

5. Scegli Crea modello.

A questo punto crea un modello di risultato. Questo modello descrive la struttura dei dati della risposta restituiti. Fa riferimento sia agli schemi di input che di output definiti nel tuoAPI.

Per creare un modello di risultato

1. Scegli Crea modello.
2. Per Nome, immetti **`result`**.
3. Per Tipo di contenuto inserisci **`application/json`**.

   Se non viene trovato alcun tipo di contenuto corrispondente, la convalida della richiesta non viene eseguita. Per utilizzare lo stesso modello indipendentemente dal tipo di contenuti, inserisci **`$default`**.
4. Per lo schema del modello, inserisci il seguente modello con il tuo *`restapi-id`*. Tuo *`restapi-id`* è elencato tra parentesi nella parte superiore della console nel seguente flusso: `API Gateway > APIs > LambdaCalc (`*`abc123`*`).`

```
{
    "type":"object",
    "properties":{
        "input":{
            "$ref":"https://apigateway.amazonaws.com/restapis/restapi-id/models/
input"
        },
        "output":{
            "$ref":"https://apigateway.amazonaws.com/restapis/restapi-id/models/
output"
        }
    },
    "title":"result"
}
```

5. Scegli Crea modello.

Ora configuri la richiesta del POST metodo per abilitare la convalida della richiesta nel corpo della richiesta in entrata.

Per abilitare la convalida della richiesta sul metodo POST

1. Nel pannello di navigazione scegli Risorse, quindi seleziona il metodo `POST` nella struttura di risorse.
2. Nella scheda Richiesta metodo, in Impostazioni richiesta metodo, scegli Modifica.
3. Per Validatore richiesta seleziona Convalida corpo.
4. Scegli Corpo della richiesta, quindi seleziona Aggiungi modello.
5. Per Tipo di contenuto inserisci **`application/json`**.

   Se non viene trovato alcun tipo di contenuto corrispondente, la convalida della richiesta non viene eseguita. Per utilizzare lo stesso modello indipendentemente dal tipo di contenuti, inserisci **`$default`**.
6. Per Modello seleziona input.
7. Seleziona Salva.

È ora possibile testare il metodo `POST` per verificare che sia stato configurato correttamente per richiamare la funzione Lambda.

Test del metodo **`POST`**

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. Per Request body, inserisci il seguente JSON payload.

   ```
   {
       "a": 1,
       "b": 2,
       "op": "+"
   }
   ```
3. Scegli Test (Esegui test).

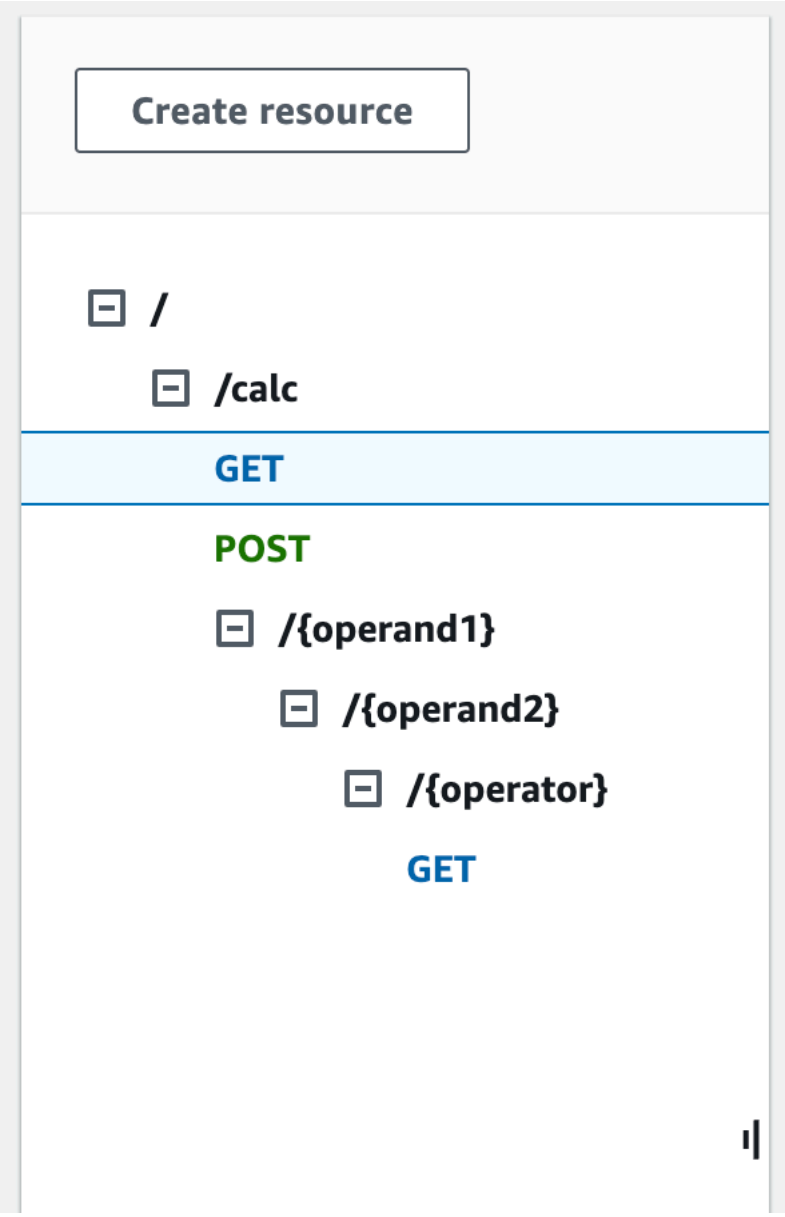
   Verrà visualizzato l'output seguente:

```
{
  "a": 1,
  "b": 2,
  "op": "+",
  "c": 3
}
```

Integrazione 3: creazione di un metodo **GET** con parametri di percorso per chiamare la funzione Lambda

Verrà ora creato un metodo `GET` in una risorsa specificata da una sequenza di parametri di percorso per chiamare la funzione Lambda di back-end. I valori dei parametri di percorso specificano i dati di input per la funzione Lambda. Verrà usato un modello di mappatura per mappare i valori dei parametri di percorso in ingresso al payload di richiesta di integrazione necessario.

La struttura API delle risorse risultante sarà simile alla seguente:



Per creare una risorsa /{operand1}/{operand2}/{operator}

1. Scegli Crea risorsa.
2. Per Percorso risorsa seleziona `/calc`.
3. Per Resource Name (Nome risorsa) immetti **`{operand1}`**.
4. Keep CORS(Cross Origin Resource Sharing) disattivato.
5. Scegli Crea risorsa.
6. Per Percorso risorsa seleziona `/calc/{operand1}/`.
7. Per Resource Name (Nome risorsa) immetti **`{operand2}`**.

8. Mantieni CORS(Cross Origin Resource Sharing) disattivato.
9. Scegli Crea risorsa.
10. Per Percorso risorsa seleziona `/calc/{operand1}/{operand2}/`.
11. Per Resource Name (Nome risorsa) immetti **`{operator}`**.
12. Mantieni CORS(Cross Origin Resource Sharing) disattivato.
13. Scegli Crea risorsa.

Questa volta utilizzerai l'integrazione Lambda integrata nella console API Gateway per configurare l'integrazione del metodo.

Per configurare l'integrazione di un metodo

1. Seleziona la risorsa /{operand1}/{operand2}/{operator}, quindi scegli Crea metodo.
2. In Tipo di metodo, seleziona GET.
3. In Tipo di integrazione, seleziona Lambda.
4. Mantieni l'opzione Integrazione proxy Lambda disattivata.
5. Per la funzione Lambda, seleziona il Regione AWS luogo in cui hai creato la funzione Lambda e inserisci. **`Calc`**
6. Mantieni attivata l'opzione Timeout predefinito.
7. Scegli Crea metodo.

Ora crei un modello di mappatura per mappare i tre parametri del URL percorso, dichiarati quando è stata creata la risorsa /calc/ {operand1}/{operand2}/{operator}, nei valori di proprietà designati nell'oggetto. JSON Poiché URL i percorsi devono essere URL codificati, l'operatore di divisione deve essere specificato come anziché. `%2F` `/` Questo modello traduce `%2F` in `'/'` prima di passarlo alla funzione Lambda.

Per creare un modello di mappatura

1. Nella scheda Richiesta di integrazione scegli Modifica in Impostazioni della richiesta di integrazione.
2. Per Richiesta corpo passthrough scegli Quando non ci sono modelli definiti (consigliato).
3. Scegli Modelli di mappatura.
4. Per Tipo di contenuto inserisci **`application/json`**.

5. Per Corpo del modello inserisci il seguente codice:

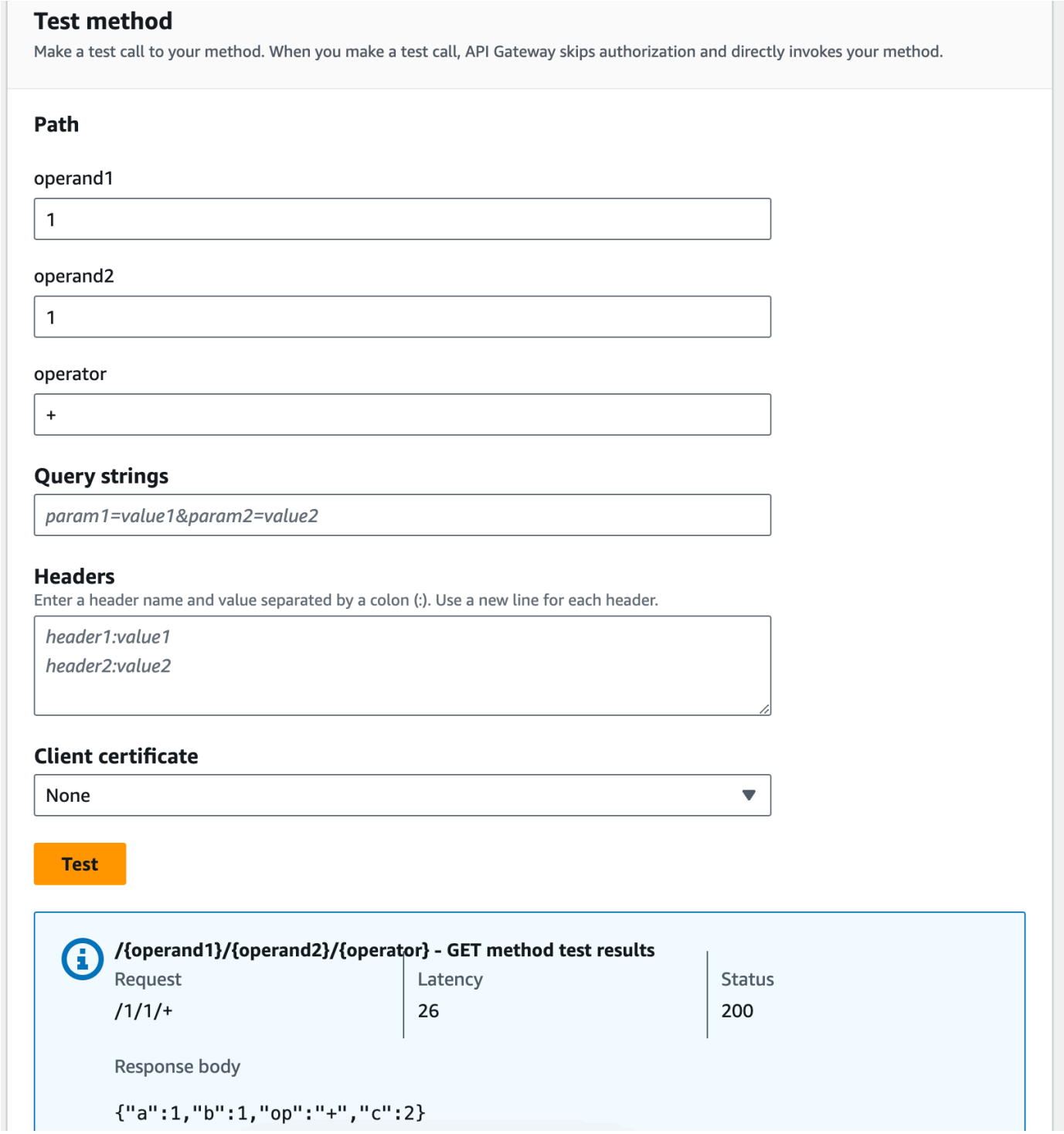
```
{
   "a": "$input.params('operand1')",
   "b": "$input.params('operand2')",
   "op":
 #if($input.params('operator')=='%2F')"/"#{else}"$input.params('operator')"#end
}
```

6. Seleziona Salva.

A questo punto testa il metodo `GET` per verificare che sia stato configurato correttamente per richiamare la funzione Lambda e passare l'output originale con la risposta di integrazione senza mappatura.

Test del metodo **`GET`**

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. Per Percorso procedi come segue:
   a. Per operand1 immetti **`1`**.
   b. Per operand2 immetti **`1`**.
   c. Per operator immetti **`+`**.
3. Scegli Test (Esegui test).
4. Il risultato deve essere simile al seguente:



A questo punto modella la struttura di dati del payload di risposta del metodo dopo lo schema `result`.

Per impostazione predefinita, al corpo della risposta del metodo viene assegnato un modello vuoto. Di conseguenza, il corpo della risposta di integrazione viene passato senza mappatura. Tuttavia,

quando si genera un file SDK per uno dei linguaggi di tipo forte, ad esempio Java o Objective-C, SDK gli utenti riceveranno come risultato un oggetto vuoto. Per garantire che sia il REST client che i SDK client ricevano il risultato desiderato, è necessario modellare i dati di risposta utilizzando uno schema predefinito. Verrà illustrato come definire un modello per il corpo della risposta del metodo e creare un modello di mappatura per convertire il corpo della risposta di integrazione nel corpo della risposta del metodo.

Per creare una risposta del metodo

1. Nella scheda Risposta metodo scegli Modifica in Risposta 200.
2. In Corpo della risposta scegli Aggiungi modello.
3. Per Tipo di contenuto inserisci **`application/json`**.
4. Per Modello seleziona risultato.
5. Seleziona Salva.

L'impostazione del modello per il corpo della risposta del metodo garantisce che i dati di risposta vengano inseriti nell'`result`oggetto di un determinato SDK metodo. Per fare in modo che i dati della risposta di integrazione siano mappati di conseguenza, è necessario un modello di mappatura.

Per creare un modello di mappatura

1. Nella scheda Risposta di integrazione scegli Modifica in Predefinito - Risposta.
2. Scegli Modelli di mappatura.
3. Per Tipo di contenuto inserisci **`application/json`**.
4. Per Corpo del modello inserisci il seguente codice:

```
#set($inputRoot = $input.path('$'))
{
  "input" : {
    "a" : $inputRoot.a,
    "b" : $inputRoot.b,
    "op" : "$inputRoot.op"
  },
  "output" : {
    "c" : $inputRoot.c
  }
}
```

5. Seleziona Salva.

Per testare il modello di mappatura

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. Per Percorso procedi come segue:
   a. Per operand1 immetti **1**.
   b. Per operand2 immetti **2**.
   c. Per operator immetti **+**.
3. Scegli Test (Esegui test).
4. Il risultato sarà simile al seguente:

```
{
  "input": {
    "a": 1,
    "b": 2,
    "op": "+"
  },
  "output": {
    "c": 3
  }
}
```

A questo punto, è possibile chiamarli solo API utilizzando la funzione Test nella console API Gateway. Per renderla disponibile ai clienti, dovrai implementare il tuoAPI. Assicurati sempre di ridistribuire il tuo API ogni volta che aggiungi, modifichi o elimini una risorsa o un metodo, aggiorni una mappatura dei dati o aggiorni le impostazioni della fase. In caso contrario, le nuove funzionalità o gli aggiornamenti non saranno disponibili per i clienti diAPI... come segue:

Per distribuire API

1. Scegliere Deploy (Implementa)API.
2. In Fase, seleziona Nuova fase.
3. In Stage name (Nome fase) immettere **`Prod`**.
4. (Facoltativo) In Description (Descrizione), immettere una descrizione.

5. Seleziona Deploy (Implementa).
6. (Facoltativo) Nella sezione Dettagli dello stage, per Invoke URL, puoi scegliere l'icona di copia per copiare API la tua chiamataURL. Puoi usarla con strumenti come Postman e c URL per testare il tuo. API

> **Note**
>
> Ridistribuisci sempre il tuo API ogni volta che aggiungi, modifichi o elimini una risorsa o un metodo, aggiorni una mappatura dei dati o aggiorni le impostazioni dello stage. In caso contrario, le nuove funzionalità o gli aggiornamenti non saranno disponibili per i tuoi clienti. API

APIDefinizioni aperte del campione API integrate con una funzione Lambda

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2017-04-20T04:08:08Z",
    "title": "LambdaCalc"
  },
  "host": "uojnr9hd57.execute-api.us-east-1.amazonaws.com",
  "basePath": "/test",
  "schemes": [
    "https"
  ],
  "paths": {
    "/calc": {
      "get": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
```

```
            "name": "operand2",
            "in": "query",
            "required": true,
            "type": "string"
          },
          {
            "name": "operator",
            "in": "query",
            "required": true,
            "type": "string"
          },
          {
            "name": "operand1",
            "in": "query",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Result"
            },
            "headers": {
              "operand_1": {
                "type": "string"
              },
              "operand_2": {
                "type": "string"
              },
              "operator": {
                "type": "string"
              }
            }
          }
        },
        "x-amazon-apigateway-request-validator": "Validate query string parameters
 and headers",
        "x-amazon-apigateway-integration": {
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "responses": {
            "default": {
              "statusCode": "200",
```

```
              "responseParameters": {
                "method.response.header.operator": "integration.response.body.op",
                "method.response.header.operand_2": "integration.response.body.b",
                "method.response.header.operand_1": "integration.response.body.a"
              },
              "responseTemplates": {
                "application/json": "#set($res = $input.path('$'))\n{\n    \"result
\": \"$res.a, $res.b, $res.op => $res.c\",\n  \"a\" : \"$res.a\",\n  \"b\" :
 \"$res.b\",\n  \"op\" : \"$res.op\",\n  \"c\" : \"$res.c\"\n}"
              }
            }
          },
          "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:Calc/invocations",
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST",
          "requestTemplates": {
            "application/json": "{\n    \"a\":  \"$input.params('operand1')\",\n
 \"b\":  \"$input.params('operand2')\", \n    \"op\": \"$input.params('operator')\"
  \n}"
          },
          "type": "aws"
        }
      },
      "post": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "in": "body",
            "name": "Input",
            "required": true,
            "schema": {
              "$ref": "#/definitions/Input"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
```

```
            "schema": {
              "$ref": "#/definitions/Result"
            }
          }
        },
        "x-amazon-apigateway-request-validator": "Validate body",
        "x-amazon-apigateway-integration": {
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "responses": {
            "default": {
              "statusCode": "200",
              "responseTemplates": {
                "application/json": "#set($inputRoot = $input.path('$'))\n{\n  \"a
\" : $inputRoot.a,\n  \"b\" : $inputRoot.b,\n  \"op\" : $inputRoot.op,\n  \"c\" :
 $inputRoot.c\n}"
              }
            }
          },
          "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:Calc/invocations",
          "passthroughBehavior": "when_no_templates",
          "httpMethod": "POST",
          "type": "aws"
        }
      }
    },
    "/calc/{operand1}/{operand2}/{operator}": {
      "get": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "operand2",
            "in": "path",
            "required": true,
            "type": "string"
          },
          {
            "name": "operator",
            "in": "path",
```

```
            "required": true,
            "type": "string"
          },
          {
            "name": "operand1",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Result"
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "responses": {
            "default": {
              "statusCode": "200",
              "responseTemplates": {
                "application/json": "#set($inputRoot = $input.path('$'))\n{\n
 \"input\" : {\n    \"a\" : $inputRoot.a,\n    \"b\" : $inputRoot.b,\n    \"op\" :
 \"$inputRoot.op\"\n  },\n  \"output\" : {\n    \"c\" : $inputRoot.c\n  }\n}"
              }
            }
          },
          "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:Calc/invocations",
          "passthroughBehavior": "when_no_templates",
          "httpMethod": "POST",
          "requestTemplates": {
            "application/json": "{\n   \"a\": \"$input.params('operand1')\",
\n   \"b\": \"$input.params('operand2')\",\n   \"op\":
 #if($input.params('operator')=='%2F')\"/\"#{else}\"$input.params('operator')\"#end
\n   \n}"
          },
          "contentHandling": "CONVERT_TO_TEXT",
          "type": "aws"
        }
      }
```

```
    }
  },
  "definitions": {
    "Input": {
      "type": "object",
      "required": [
        "a",
        "b",
        "op"
      ],
      "properties": {
        "a": {
          "type": "number"
        },
        "b": {
          "type": "number"
        },
        "op": {
          "type": "string",
          "description": "binary op of ['+', 'add', '-', 'sub', '*', 'mul', '%2F',
'div']"
        }
      },
      "title": "Input"
    },
    "Output": {
      "type": "object",
      "properties": {
        "c": {
          "type": "number"
        }
      },
      "title": "Output"
    },
    "Result": {
      "type": "object",
      "properties": {
        "input": {
          "$ref": "#/definitions/Input"
        },
        "output": {
          "$ref": "#/definitions/Output"
        }
      },
```

```
      "title": "Result"
    }
  },
  "x-amazon-apigateway-request-validators": {
    "Validate body": {
      "validateRequestParameters": false,
      "validateRequestBody": true
    },
    "Validate query string parameters and headers": {
      "validateRequestParameters": true,
      "validateRequestBody": false
    }
  }
}
```

Tutorial: creare un REST API proxy come Amazon S3

Come esempio per illustrare l'utilizzo di un REST API in API Gateway per eseguire il proxy di Amazon S3, questa sezione descrive come creare e configurare REST API un gateway per esporre le seguenti operazioni di Amazon S3:

- Esporre GET sulla API risorsa root per elencare tutti i bucket Amazon S3 di un chiamante.
- Esporre GET su una risorsa Folder per visualizzare un elenco di tutti gli oggetti in un bucket Amazon S3.
- Esporre GET su una risorsa Folder/Item per visualizzare o scaricare un oggetto da un bucket Amazon S3.

Potresti voler importare l'esempio API come proxy Amazon S3, come mostrato in. APIDefinizioni aperte dell'esempio API come proxy Amazon S3 Questo esempio include più metodi esposti. Per istruzioni su come importare e API utilizzare la API definizione Open, consultaSviluppa REST APIs utilizzando Open API in API Gateway.

> **Note**
>
> Per integrare il tuo API Gateway API con Amazon S3, devi scegliere una regione in cui siano disponibili sia i servizi API Gateway che Amazon S3. Per la disponibilità regionale, consulta Amazon API Gateway Endpoints and Quotas.

Argomenti

- Imposta IAM le autorizzazioni per API richiamare le azioni Amazon S3
- Crea API risorse per rappresentare le risorse di Amazon S3
- Esporre un API metodo per elencare i bucket Amazon S3 del chiamante
- Mostra API i metodi per accedere a un bucket Amazon S3
- Esponi API i metodi per accedere a un oggetto Amazon S3 in un bucket
- APIDefinizioni aperte dell'esempio API come proxy Amazon S3
- Chiamalo API usando un client REST API

Imposta IAM le autorizzazioni per API richiamare le azioni Amazon S3

Per consentire loro di API richiamare le azioni di Amazon S3, è necessario che le politiche IAM appropriate siano associate a IAM un ruolo.

Per creare il ruolo di esecuzione del proxy AWS di servizio

1. Accedi a AWS Management Console e apri la IAM console all'indirizzo https://console.aws.amazon.com/iam/.
2. Scegliere Roles (Ruoli).
3. Scegliere Crea ruolo.
4. Scegli il AWS servizio in Seleziona il tipo di entità affidabile, quindi seleziona APIGateway e seleziona Consenti a API Gateway di inviare i log ai CloudWatch log.
5. Scegli Successivo e di nuovo Successivo.
6. In Role name (Nome ruolo) immettere **`APIGatewayS3ProxyPolicy`** e quindi selezionare Create role (Crea ruolo).
7. Nell'elenco Roles (Ruoli) scegliere il ruolo appena creato. Potrebbe essere necessario scorrere o utilizzare la barra di ricerca per trovare il ruolo.
8. Per il ruolo selezionato, seleziona la scheda Aggiungi autorizzazioni.
9. Dall'elenco a discesa scegli Collega policy.
10. Nella barra di ricerca inserisci **`AmazonS3FullAccess`** e scegli Aggiungi autorizzazioni.

> **Note**
>
> Per semplicità, questo tutorial utilizza una policy gestita. Come procedura ottimale, è consigliabile creare una IAM politica personalizzata per concedere le autorizzazioni minime richieste.

11. Nota il ruolo appena creatoARN, lo utilizzerai in seguito.

Crea API risorse per rappresentare le risorse di Amazon S3

Utilizzi API la risorsa root (`/`) come contenitore dei bucket Amazon S3 di un chiamante autenticato. È inoltre possibile creare `Item` risorse `Folder` e per rappresentare rispettivamente un particolare bucket Amazon S3 e un particolare oggetto Amazon S3. Il nome della cartella e la chiave dell'oggetto verranno specificati, sotto forma di parametri di percorso come parte di una richiestaURL, dal chiamante.

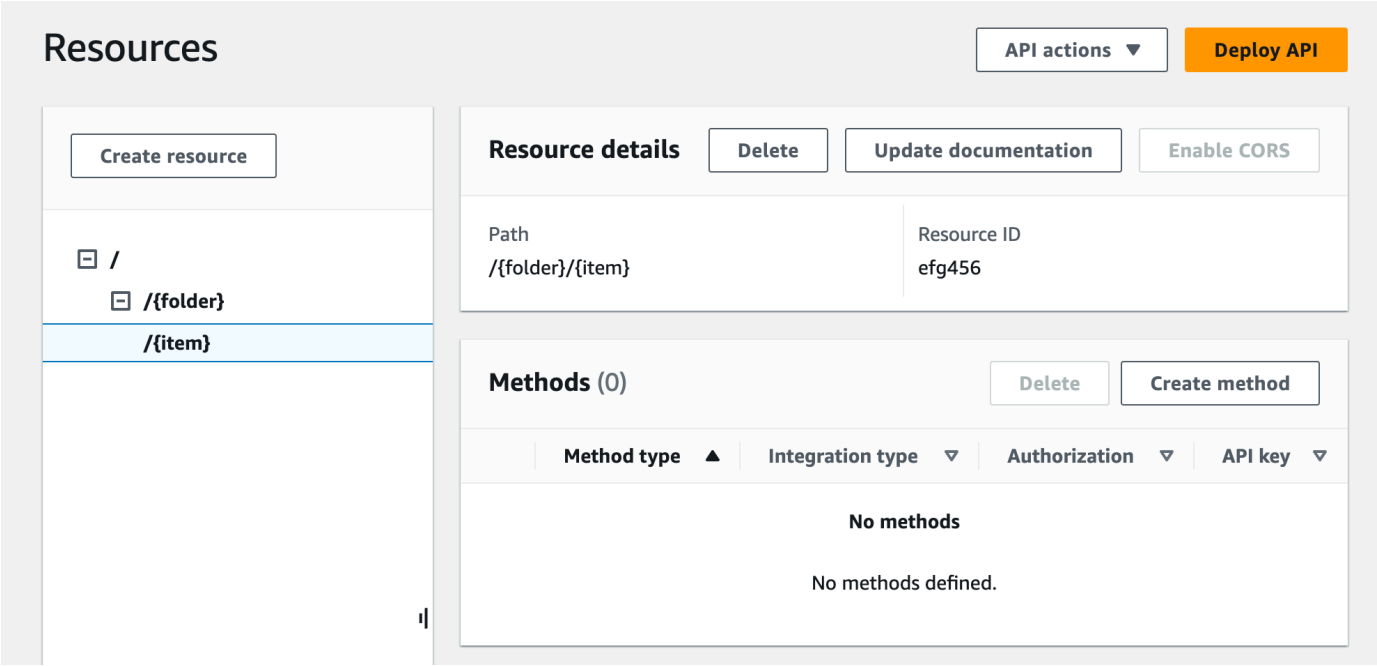
> **Note**
>
> Quando si accede a oggetti la cui chiave oggetto include `/` o qualsiasi altro carattere speciale, il carattere deve essere URL codificato. Ad esempio, `test/test.txt` dovrebbe avere la codifica `test%2Ftest.txt`.

Per creare una API risorsa che esponga le caratteristiche del servizio Amazon S3

1. Nello stesso modo in Regione AWS cui hai creato il tuo bucket Amazon S3, crea un API bucket chiamato MyS3. Questa API risorsa principale (/) rappresenta il servizio Amazon S3. In questa fase crei le due risorse aggiuntive /{folder} e /{item}.
2. Scegli Crea risorsa.
3. Mantieni l'opzione Risorsa proxy disattivata.
4. Per Percorso risorsa seleziona `/`.
5. Per Resource Name (Nome risorsa) immetti **`{folder}`**.
6. Mantieni CORS(Cross Origin Resource Sharing) deselezionato.
7. Scegli Crea risorsa.
8. Seleziona la risorsa /{folder}, quindi scegli Crea risorsa.

9. Ripeti le fasi precedenti per creare una risorsa figlio /{folder} denominata **`{item}`**.

   Il tuo finale API dovrebbe essere simile al seguente:



Esporre un API metodo per elencare i bucket Amazon S3 del chiamante

Per ottenere l'elenco dei bucket Amazon S3 del chiamante è necessario richiamare l'azione di servizio GETsu Amazon S3. Sulla risorsa principale (/), crea il metodo. API GET Configura il GET metodo di integrazione con Amazon S3 come segue.

Per creare e inizializzare il metodo's API **`GET /`**

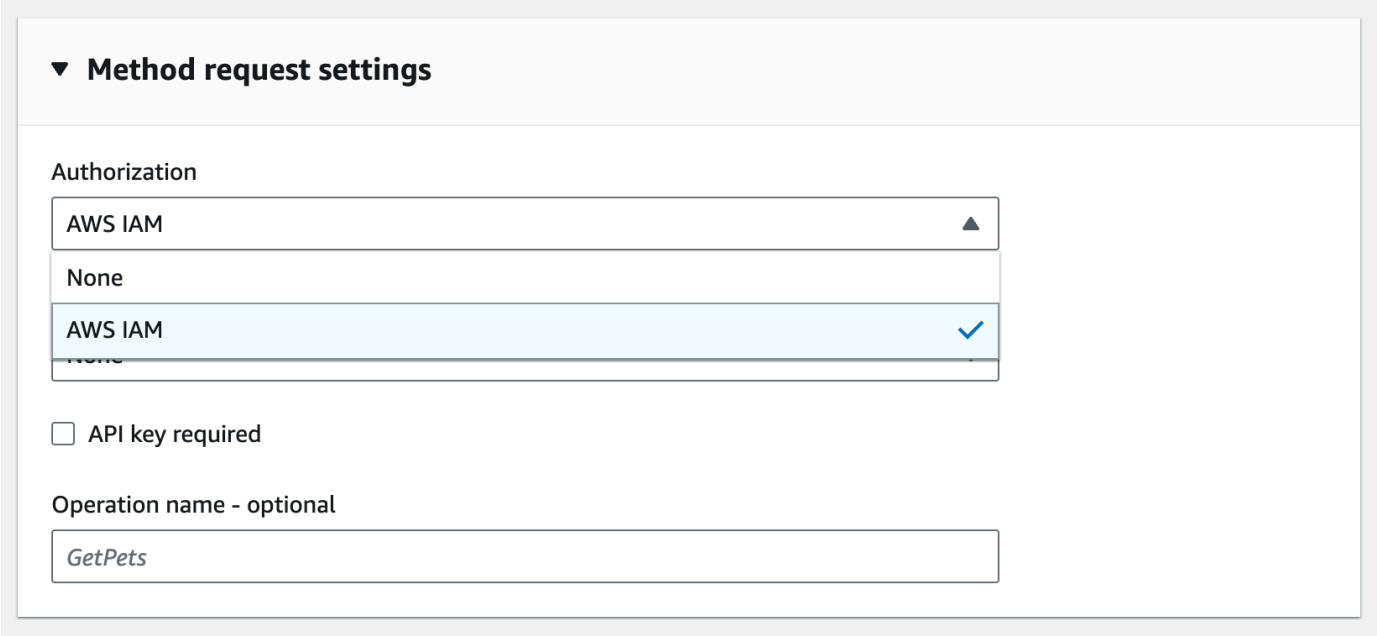
1. Seleziona la risorsa /, quindi scegli Crea metodo.
2. Per il tipo di metodo, selezionare GET.
3. Per Tipo di integrazione seleziona Servizio AWS.
4. Per Regione AWS, seleziona il Regione AWS luogo in cui hai creato il tuo bucket Amazon S3.
5. Per Servizio AWS seleziona Amazon Simple Storage Service.
6. Lascia vuoto Sottodominio AWS .
7. Per il HTTPmetodo, seleziona. GET
8. In Tipo di operazione scegli Utilizza sostituzione percorso.

Con path override, API Gateway inoltra la richiesta del client ad Amazon S3 come richiesta corrispondente in stile percorso Amazon REST API S3, in cui una risorsa Amazon S3 è espressa dal percorso di risorsa del pattern. `s3-host-name/bucket/key` APIGateway imposta `s3-host-name` e passa il client specificato `bucket` e `key` dal client ad Amazon S3.

9. In Sostituzione percorso immetti /.
10. Per il ruolo di esecuzione, inserisci il ruolo ARN per**`APIGatewayS3ProxyPolicy`**.
11. Scegliete le impostazioni di richiesta del metodo.

    Utilizzi le impostazioni di richiesta del metodo per controllare chi può chiamare questo tuo metodoAPI.

12. Per Autorizzazioni seleziona `AWS_IAM` nel menu a discesa.



13. Scegli Crea metodo.

Questa configurazione integra la richiesta `GET https://`*`your-api-host`*`/`*`stage`*`/` front-end con il back-end `GET https://`*`your-s3-host`*`/`.

APIPer restituire correttamente le risposte e le eccezioni corrette al chiamante, è necessario dichiarare le 200, 400 e 500 risposte in Method response. Si utilizza la mappatura predefinita per 200 risposte in modo che le risposte di backend con il codice di stato non dichiarato qui vengano restituite al chiamante come 200 risposte.

Per dichiarare i tipi di risposta del metodo **`GET /`**

1. Nella scheda Risposta metodo scegli Modifica in Risposta 200.
2. Scegli Aggiungi intestazione e procedi come descritto di seguito:

   a. Per Nome intestazione immetti **`Content-Type`**.

   b. Seleziona Add header (Aggiungi intestazione).

   Ripeti queste fasi per creare un'intestazione **`Timestamp`** e un'intestazione **`Content-Length`**.
3. Seleziona Salva.
4. Nella scheda Risposta metodo scegli Crea risposta in Risposte del metodo.
5. Per il codice HTTP di stato, inserisci 400.

   Non impostare alcuna intestazione per questa risposta.
6. Seleziona Salva.
7. Ripeti le seguenti fasi per creare la risposta 500.

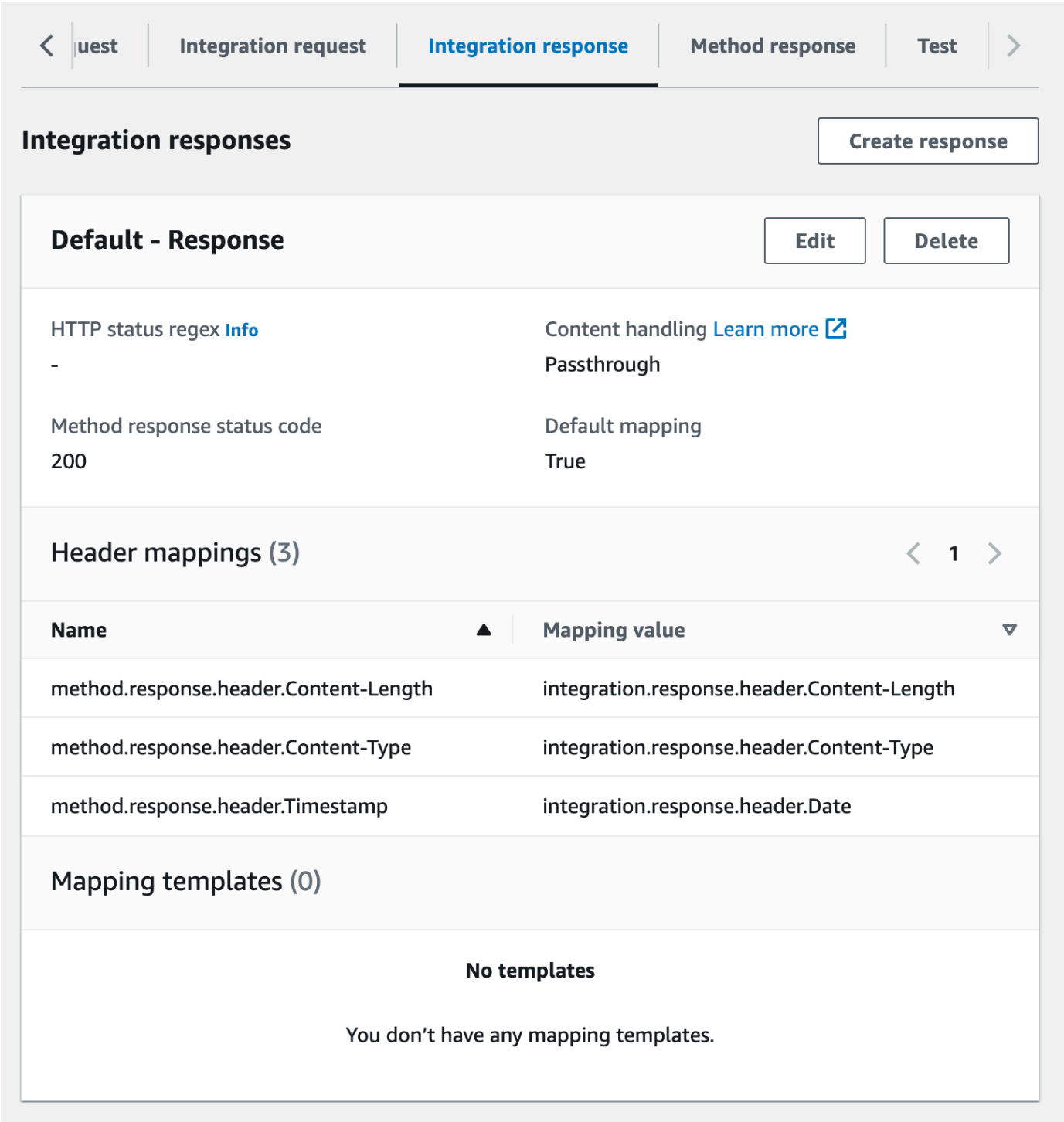
   Non impostare alcuna intestazione per questa risposta.

Poiché la risposta di integrazione corretta di Amazon S3 restituisce la bucket list come XML payload e la risposta del metodo predefinito di API Gateway restituisce un JSON payload, è necessario mappare il valore del parametro di intestazione Content-Type del backend alla controparte frontend. In caso contrario, il client riceverà `application/json` per il tipo di contenuto quando il corpo della risposta è in realtà una stringa. XML La procedura seguente descrive come eseguire questa configurazione. Inoltre, desideri visualizzare al client anche altri parametri di intestazione, come Date e Content-Length.

Per impostare le mappature delle intestazioni di risposta per il metodo/GET

1. Nella scheda Risposta di integrazione scegli Modifica in Predefinito - Risposta.
2. Per l'intestazione Content-Length immetti **`integration.response.header.Content-Length`** come valore di mappatura.
3. Per l'intestazione Content-Type immetti **`integration.response.header.Content-Type`** come valore di mappatura.
4. Per l'intestazione Timestamp immetti **`integration.response.header.Date`** come valore di mappatura.

5. Seleziona Salva. Il risultato sarà essere simile al seguente:



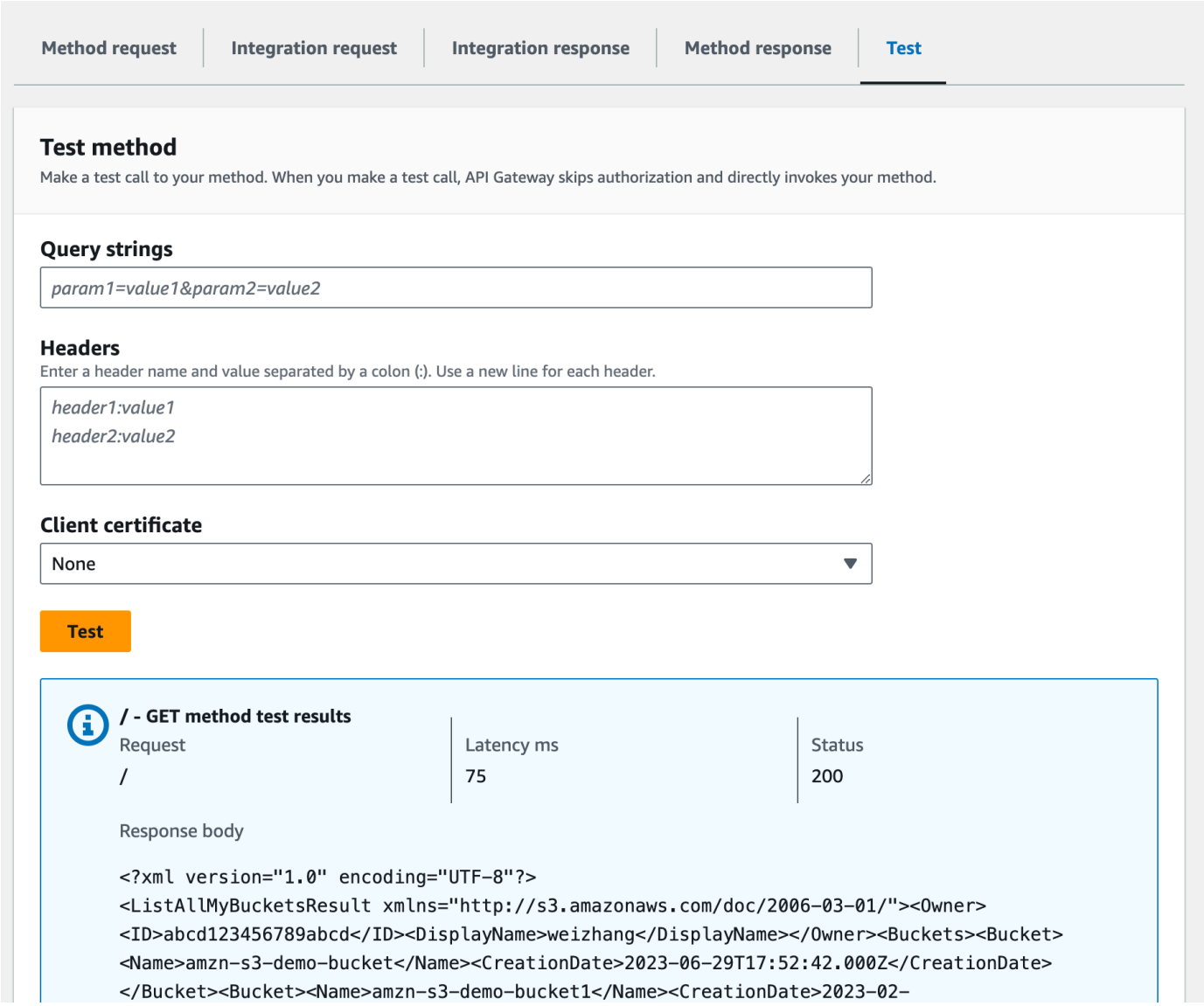
6. Nella scheda Risposta di integrazione scegli Crea risposta in Risposte di integrazione.
7. Per l'espressione regolare HTTP dello stato, immettere. **`4\d{2}`** Questo associa tutti i codici di stato HTTP della risposta 4xx alla risposta del metodo.

8. Per Codice di stato risposta metodo seleziona **`400`**.
9. Scegli Create (Crea) .
10. Ripeti le seguenti fasi per creare una risposta di integrazione per la risposta del metodo 500. Per l'espressione regolare HTTP dello stato, immettere. **`5\d{2}`**

Come buona pratica, puoi testare quello API che hai configurato finora.

Test del metodo **`GET /`**

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. Scegli Test (Esegui test). Il risultato sarà simile all'immagine seguente:



Mostra API i metodi per accedere a un bucket Amazon S3

Per lavorare con un bucket Amazon S3, esponi il `GET` metodo sulla risorsa/{folder} per elencare gli oggetti in un bucket. Le istruzioni sono simili a quelle descritte in Esporre un API metodo per elencare i bucket Amazon S3 del chiamante. Per altri metodi, puoi importare l'esempio qui,. API APIDefinizioni aperte dell'esempio API come proxy Amazon S3

Per esporre il GET metodo su una risorsa di cartella

1. Seleziona la risorsa /{folder}, quindi scegli Crea metodo.
2. Per il tipo di metodo, selezionare GET.

3. Per Tipo di integrazione seleziona Servizio AWS.
4. Per Regione AWS, seleziona il Regione AWS luogo in cui hai creato il tuo bucket Amazon S3.
5. Per Servizio AWS seleziona Amazon Simple Storage Service.
6. Lascia vuoto Sottodominio AWS .
7. Per il HTTPmetodo, seleziona. GET
8. In Tipo di operazione scegli Utilizza sostituzione percorso.
9. Per Sostituzione percorso immetti **`{bucket}`**.
10. Per Ruolo di esecuzione, immettere il ruolo ARN per**`APIGatewayS3ProxyPolicy`**.
11. Scegli Crea metodo.

Il parametro `{folder}` path viene impostato nell'endpoint Amazon S3. URL Dovrai mappare il parametro di percorso `{folder}` della richiesta di metodo al parametro di percorso `{bucket}` della richiesta di integrazione.

Per mappare **`{folder}`** a **`{bucket}`**

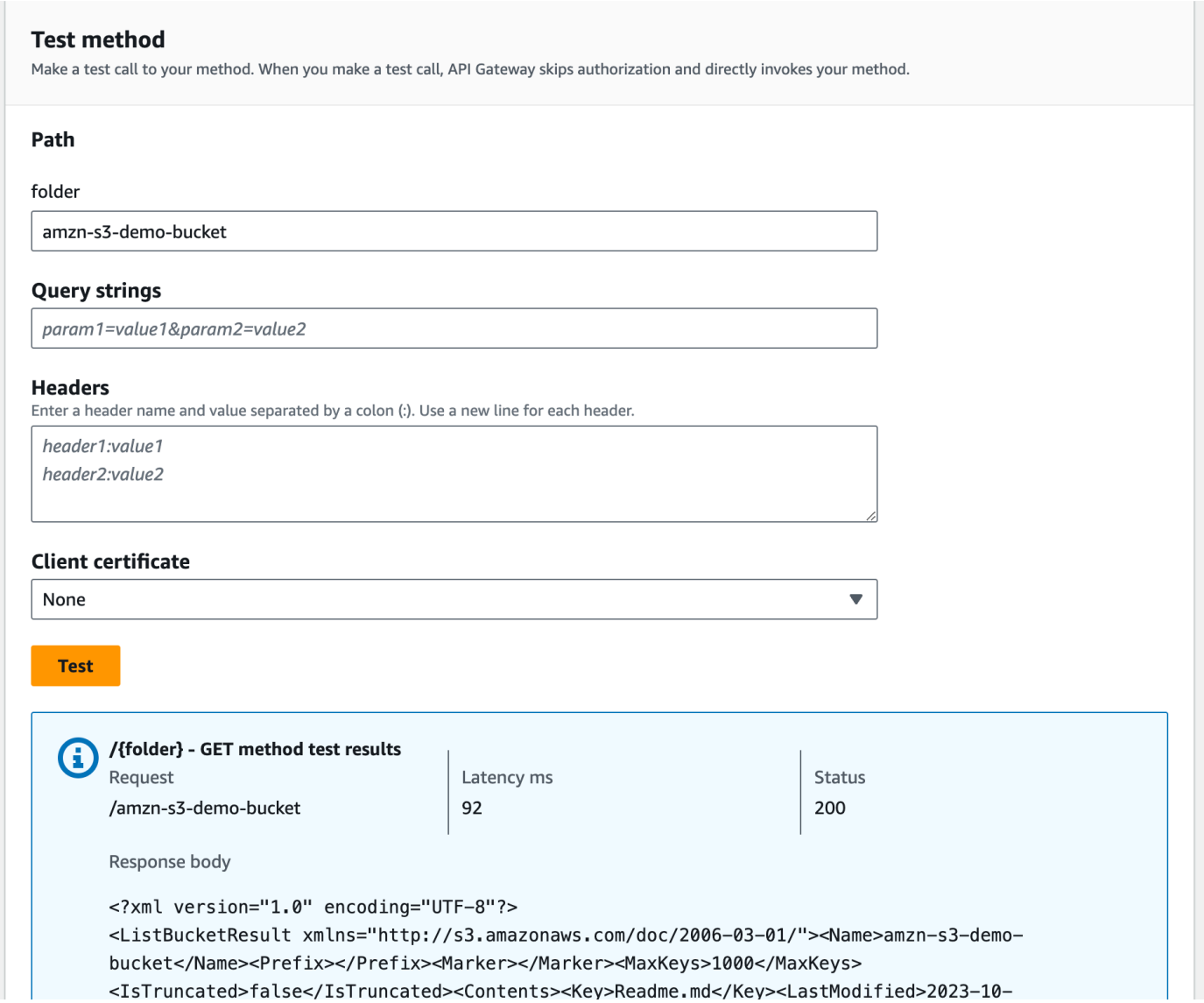
1. Nella scheda Richiesta di integrazione scegli Modifica in Impostazioni della richiesta di integrazione.
2. Scegli i parametri del URL percorso, quindi scegli Aggiungi parametro di percorso.
3. Per Nome, immetti **`bucket`**.
4. In Mappato da, inserire **`method.request.path.folder`**.
5. Seleziona Salva.

Ora metti alla prova il tuoAPI.

Per testare il metodo **`/{folder} GET`**

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. In Percorso immetti il nome del tuo bucket per folder.
3. Scegli Test (Esegui test).

   Il risultato del test conterrà l'elenco degli oggetti presenti nel bucket.



Esponi API i metodi per accedere a un oggetto Amazon S3 in un bucket

Amazon S3 supportaGET,, DELETE HEADOPTIONS, POST e PUT azioni per accedere e gestire gli oggetti in un determinato bucket. In questo tutorial, esponi un metodo `GET` nella risorsa `{folder}/{item}` per ottenere un'immagine da un bucket. Per ulteriori applicazioni della `{folder}/{item}` risorsa, consulta l'esempioAPI,. APIDefinizioni aperte dell'esempio API come proxy Amazon S3

Per esporre il GET metodo su un elemento (risorsa)

1. Seleziona la risorsa /{item}, quindi scegli Crea metodo.
2. Per il tipo di metodo, selezionare GET.

3. Per Tipo di integrazione seleziona Servizio AWS.
4. Per Regione AWS, seleziona il Regione AWS luogo in cui hai creato il tuo bucket Amazon S3.
5. Per Servizio AWS seleziona Amazon Simple Storage Service.
6. Lascia vuoto Sottodominio AWS .
7. Per il HTTPmetodo, seleziona. GET
8. In Tipo di operazione scegli Utilizza sostituzione percorso.
9. Per Sostituzione percorso immetti {bucket}/{object}.
10. Per Ruolo di esecuzione, immettere il ruolo ARN per**`APIGatewayS3ProxyPolicy`**.
11. Scegli Crea metodo.

I parametri `{folder}` e il `{item}` percorso vengono impostati nell'endpoint Amazon S3. URL Dovrai mappare il parametro di percorso della richiesta di metodo al parametro di percorso della richiesta di integrazione.

In questa fase si effettuano le operazioni seguenti:

- Mappa il parametro di percorso `{folder}` della richiesta di metodo al parametro di percorso `{bucket}` della richiesta di integrazione.
- Mappa il parametro di percorso `{item}` della richiesta di metodo al parametro di percorso `{object}` della richiesta di integrazione.

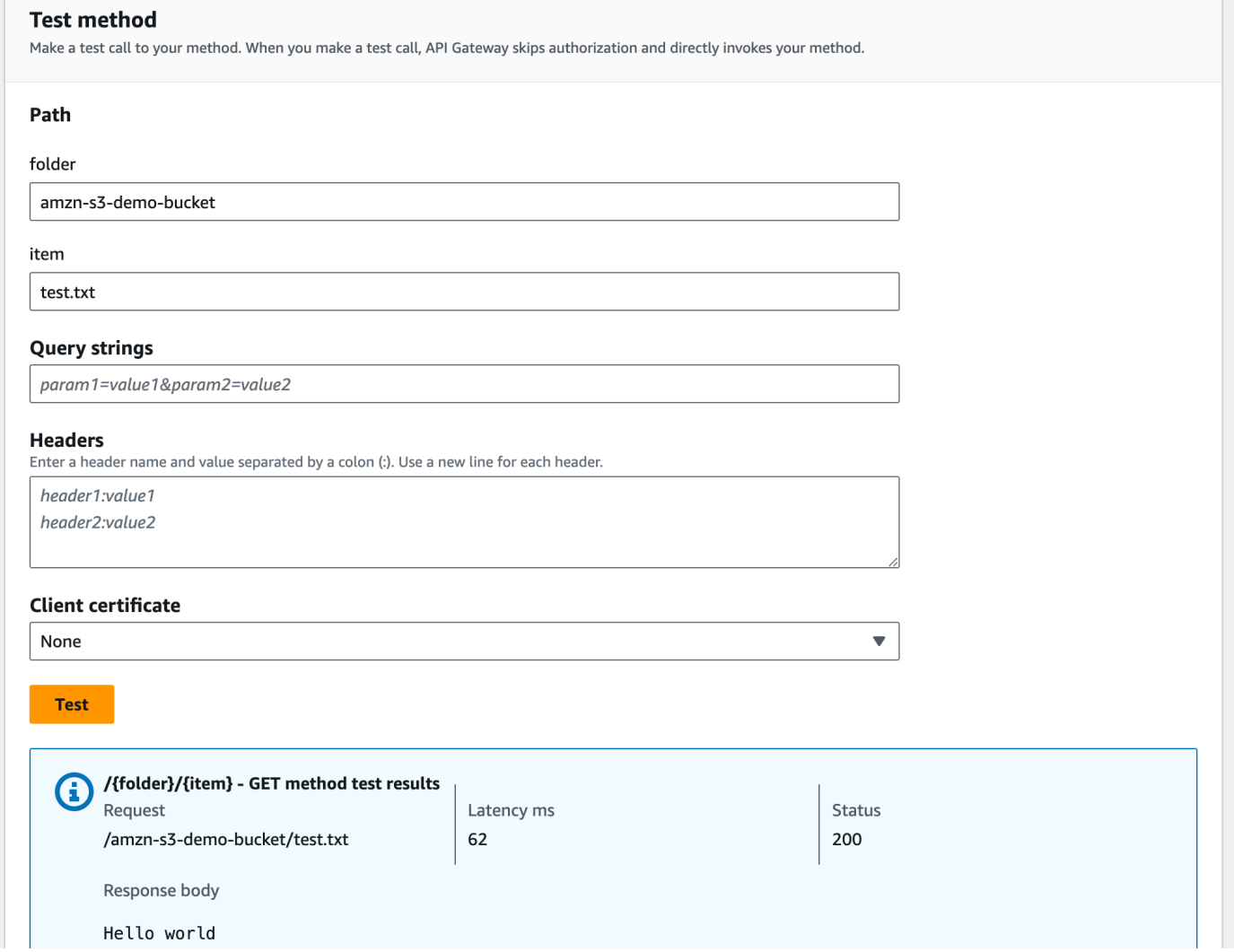
Per mappare **`{folder}`** a **`{bucket}`** e **`{item}`** a **`{object}`**

1. Nella scheda Richiesta di integrazione scegli Modifica in Impostazioni della richiesta di integrazione.
2. Scegli i parametri del URLpercorso.
3. Scegli Aggiungi parametro di percorso.
4. Per Nome, immetti **`bucket`**.
5. In Mappato da, inserire **`method.request.path.folder`**.
6. Scegli Aggiungi parametro di percorso.
7. Per Nome, immetti **`object`**.
8. In Mappato da, inserire **`method.request.path.item`**.
9. Seleziona Salva.

Per testare il metodo **`/{folder}/{object} GET`**

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. In Percorso immetti il nome del tuo bucket per folder.
3. In Percorso immetti il nome di un elemento per item.
4. Scegli Test (Esegui test).

   Il corpo della risposta conterrà il contenuto dell'elemento.



La richiesta restituisce correttamente il testo semplice di («Hello world») come contenuto del file specificato (test.txt) nel bucket Amazon S3 specificato (amzn-s3-demo-bucket).

Per scaricare o caricare file binari, che in API Gateway sono considerati qualsiasi cosa diversa dal contenuto con codifica utf-8, sono necessarie impostazioni aggiuntive. JSON API come mostrato di seguito:

Per scaricare o caricare file binari da S3

1. Registra i tipi di file multimediali del file interessato su. API binaryMediaTypes Questa operazione può essere eseguita nella console:
   a. Scegli APIIle impostazioni perAPI.
   b. In Tipi di media binari scegli Gestisci tipi di supporti.
   c. Scegli Aggiungi tipo di supporto binario, quindi immetti il tipo di supporto richiesto, ad esempio `image/png`.
   d. Scegli Salva modifiche per salvare l'impostazione.
2. Aggiungi l'intestazione `Content-Type` (per il caricamento) e/o `Accept` (per il download) alla richiesta di metodo per chiedere al client di specificare il tipo di supporto binario richiesto e mapparlo alla richiesta di integrazione.
3. Imposta Content Handling (Gestione contenuto) su `Passthrough` nella richiesta di integrazione (per il caricamento) e in una risposta di integrazione (per il download). Assicurati che per il tipo di contenuto interessato non sia definito un modello di mappatura. Per ulteriori informazioni, consulta Integration Passthrough Behaviors e Select VTL Mapping Templates.

Il limite della dimensione del payload è 10 MB. Consulta APIQuote del gateway per la configurazione e l'esecuzione di un REST API.

Assicurati che ai file in Amazon S3 vengano aggiunti tipi di contenuto corretti come metadati dei file. Per il contenuto multimediale riproducibile in streaming, potrebbe essere necessario aggiungere anche `Content-Disposition:inline` ai metadati.

Per ulteriori informazioni sul supporto binario in API Gateway, consulta. Conversioni dei tipi di contenuto in Gateway API

APIDefinizioni aperte dell'esempio API come proxy Amazon S3

Le seguenti API definizioni aperte descrivono un API proxy che funziona come proxy Amazon S3. APIContiene più operazioni Amazon S3 rispetto a quelle che API hai creato nel tutorial. I seguenti metodi sono esposti nelle API definizioni Open:

- Esporre GET sulla API risorsa root per elencare tutti i bucket Amazon S3 di un chiamante.
- Esporre GET su una risorsa Folder per visualizzare un elenco di tutti gli oggetti in un bucket Amazon S3.
- Esporre PUT su una risorsa Folder per aggiungere un bucket ad Amazon S3.
- Esporre DELETE su una risorsa Folder per rimuovere un bucket da Amazon S3.
- Esporre GET su una risorsa Folder/Item per visualizzare o scaricare un oggetto da un bucket Amazon S3.
- Esporre PUT su una risorsa Folder/Item per caricare un oggetto in un bucket Amazon S3.
- Esporre HEAD su una risorsa Folder/Item per ottenere i metadati degli oggetti in un bucket Amazon S3.
- Esporre DELETE su una risorsa Folder/Item per rimuovere un oggetto da un bucket Amazon S3.

Per istruzioni su come importare e API utilizzare la definizione Open, consulta. API Sviluppa REST APIs utilizzando Open API in API Gateway

Per istruzioni su come creare un file simileAPI, vedereTutorial: creare un REST API proxy come Amazon S3.

Per informazioni su come richiamarlo API utilizzando Postman, che supporta l' AWS IAMautorizzazione, consulta. Chiamalo API usando un client REST API

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2016-10-13T23:04:43Z",
    "title": "MyS3"
  },
  "host": "9gn28ca086.execute-api.{region}.amazonaws.com",
  "basePath": "/S3",
  "schemes": [
    "https"
  ],
  "paths": {
    "/": {
      "get": {
        "produces": [
          "application/json"
```

```
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            },
            "headers": {
              "Content-Length": {
                "type": "string"
              },
              "Timestamp": {
                "type": "string"
              },
              "Content-Type": {
                "type": "string"
              }
            }
          },
          "400": {
            "description": "400 response"
          },
          "500": {
            "description": "500 response"
          }
        },
        "security": [
          {
            "sigv4": []
          }
        ],
        "x-amazon-apigateway-integration": {
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "responses": {
            "4\\d{2}": {
              "statusCode": "400"
            },
            "default": {
              "statusCode": "200",
              "responseParameters": {
                "method.response.header.Content-Type":
"integration.response.header.Content-Type",
                "method.response.header.Content-Length":
"integration.response.header.Content-Length",
```

```
                "method.response.header.Timestamp":
"integration.response.header.Date"
              }
            },
            "5\\d{2}": {
              "statusCode": "500"
            }
          },
          "uri": "arn:aws:apigateway:us-west-2:s3:path//",
          "passthroughBehavior": "when_no_match",
          "httpMethod": "GET",
          "type": "aws"
        }
      }
    },
    "/{folder}": {
      "get": {
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "folder",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            },
            "headers": {
              "Content-Length": {
                "type": "string"
              },
              "Date": {
                "type": "string"
              },
              "Content-Type": {
                "type": "string"
              }
```

```
            }
          },
          "400": {
            "description": "400 response"
          },
          "500": {
            "description": "500 response"
          }
        },
        "security": [
          {
            "sigv4": []
          }
        ],
        "x-amazon-apigateway-integration": {
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "responses": {
            "4\\d{2}": {
              "statusCode": "400"
            },
            "default": {
              "statusCode": "200",
              "responseParameters": {
                "method.response.header.Content-Type":
"integration.response.header.Content-Type",
                "method.response.header.Date": "integration.response.header.Date",
                "method.response.header.Content-Length":
"integration.response.header.content-length"
              }
            },
            "5\\d{2}": {
              "statusCode": "500"
            }
          },
          "requestParameters": {
            "integration.request.path.bucket": "method.request.path.folder"
          },
          "uri": "arn:aws:apigateway:us-west-2:s3:path/{bucket}",
          "passthroughBehavior": "when_no_match",
          "httpMethod": "GET",
          "type": "aws"
        }
      },
      "put": {
```

```
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "Content-Type",
            "in": "header",
            "required": false,
            "type": "string"
          },
          {
            "name": "folder",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            },
            "headers": {
              "Content-Length": {
                "type": "string"
              },
              "Content-Type": {
                "type": "string"
              }
            }
          },
          "400": {
            "description": "400 response"
          },
          "500": {
            "description": "500 response"
          }
        },
        "security": [
          {
            "sigv4": []
          }
        ],
```

```
        "x-amazon-apigateway-integration": {
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "responses": {
            "4\\d{2}": {
              "statusCode": "400"
            },
            "default": {
              "statusCode": "200",
              "responseParameters": {
                "method.response.header.Content-Type":
"integration.response.header.Content-Type",
                "method.response.header.Content-Length":
"integration.response.header.Content-Length"
              }
            },
            "5\\d{2}": {
              "statusCode": "500"
            }
          },
          "requestParameters": {
            "integration.request.path.bucket": "method.request.path.folder",
            "integration.request.header.Content-Type":
"method.request.header.Content-Type"
          },
          "uri": "arn:aws:apigateway:us-west-2:s3:path/{bucket}",
          "passthroughBehavior": "when_no_match",
          "httpMethod": "PUT",
          "type": "aws"
        }
      },
      "delete": {
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "folder",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
```

```
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            },
            "headers": {
              "Date": {
                "type": "string"
              },
              "Content-Type": {
                "type": "string"
              }
            }
          },
          "400": {
            "description": "400 response"
          },
          "500": {
            "description": "500 response"
          }
        },
        "security": [
          {
            "sigv4": []
          }
        ],
        "x-amazon-apigateway-integration": {
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "responses": {
            "4\\d{2}": {
              "statusCode": "400"
            },
            "default": {
              "statusCode": "200",
              "responseParameters": {
                "method.response.header.Content-Type":
 "integration.response.header.Content-Type",
                "method.response.header.Date": "integration.response.header.Date"
              }
            },
            "5\\d{2}": {
              "statusCode": "500"
            }
          },
          "requestParameters": {
```

```
            "integration.request.path.bucket": "method.request.path.folder"
          },
          "uri": "arn:aws:apigateway:us-west-2:s3:path/{bucket}",
          "passthroughBehavior": "when_no_match",
          "httpMethod": "DELETE",
          "type": "aws"
        }
      }
    },
    "/{folder}/{item}": {
      "get": {
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "item",
            "in": "path",
            "required": true,
            "type": "string"
          },
          {
            "name": "folder",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            },
            "headers": {
              "content-type": {
                "type": "string"
              },
              "Content-Type": {
                "type": "string"
              }
            }
          },
          "400": {
```

```
            "description": "400 response"
          },
          "500": {
            "description": "500 response"
          }
        },
        "security": [
          {
            "sigv4": []
          }
        ],
        "x-amazon-apigateway-integration": {
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "responses": {
            "4\\d{2}": {
              "statusCode": "400"
            },
            "default": {
              "statusCode": "200",
              "responseParameters": {
                "method.response.header.content-type":
"integration.response.header.content-type",
                "method.response.header.Content-Type":
"integration.response.header.Content-Type"
              }
            },
            "5\\d{2}": {
              "statusCode": "500"
            }
          },
          "requestParameters": {
            "integration.request.path.object": "method.request.path.item",
            "integration.request.path.bucket": "method.request.path.folder"
          },
          "uri": "arn:aws:apigateway:us-west-2:s3:path/{bucket}/{object}",
          "passthroughBehavior": "when_no_match",
          "httpMethod": "GET",
          "type": "aws"
        }
      },
      "head": {
        "produces": [
          "application/json"
        ],
```

```
        "parameters": [
          {
            "name": "item",
            "in": "path",
            "required": true,
            "type": "string"
          },
          {
            "name": "folder",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            },
            "headers": {
              "Content-Length": {
                "type": "string"
              },
              "Content-Type": {
                "type": "string"
              }
            }
          },
          "400": {
            "description": "400 response"
          },
          "500": {
            "description": "500 response"
          }
        },
        "security": [
          {
            "sigv4": []
          }
        ],
        "x-amazon-apigateway-integration": {
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "responses": {
```

```
            "4\\d{2}": {
              "statusCode": "400"
            },
            "default": {
              "statusCode": "200",
              "responseParameters": {
                "method.response.header.Content-Type":
"integration.response.header.Content-Type",
                "method.response.header.Content-Length":
"integration.response.header.Content-Length"
              }
            },
            "5\\d{2}": {
              "statusCode": "500"
            }
          },
          "requestParameters": {
            "integration.request.path.object": "method.request.path.item",
            "integration.request.path.bucket": "method.request.path.folder"
          },
          "uri": "arn:aws:apigateway:us-west-2:s3:path/{bucket}/{object}",
          "passthroughBehavior": "when_no_match",
          "httpMethod": "HEAD",
          "type": "aws"
        }
      },
      "put": {
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "Content-Type",
            "in": "header",
            "required": false,
            "type": "string"
          },
          {
            "name": "item",
            "in": "path",
            "required": true,
            "type": "string"
          },
          {
```

```
            "name": "folder",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            },
            "headers": {
              "Content-Length": {
                "type": "string"
              },
              "Content-Type": {
                "type": "string"
              }
            }
          },
          "400": {
            "description": "400 response"
          },
          "500": {
            "description": "500 response"
          }
        },
        "security": [
          {
            "sigv4": []
          }
        ],
        "x-amazon-apigateway-integration": {
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "responses": {
            "4\\d{2}": {
              "statusCode": "400"
            },
            "default": {
              "statusCode": "200",
              "responseParameters": {
                "method.response.header.Content-Type":
 "integration.response.header.Content-Type",
```

```
                "method.response.header.Content-Length":
"integration.response.header.Content-Length"
              }
            },
            "5\\d{2}": {
              "statusCode": "500"
            }
          },
          "requestParameters": {
            "integration.request.path.object": "method.request.path.item",
            "integration.request.path.bucket": "method.request.path.folder",
            "integration.request.header.Content-Type":
"method.request.header.Content-Type"
          },
          "uri": "arn:aws:apigateway:us-west-2:s3:path/{bucket}/{object}",
          "passthroughBehavior": "when_no_match",
          "httpMethod": "PUT",
          "type": "aws"
        }
      },
      "delete": {
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "item",
            "in": "path",
            "required": true,
            "type": "string"
          },
          {
            "name": "folder",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            },
```

```
            "headers": {
              "Content-Length": {
                "type": "string"
              },
              "Content-Type": {
                "type": "string"
              }
            }
          },
          "400": {
            "description": "400 response"
          },
          "500": {
            "description": "500 response"
          }
        },
        "security": [
          {
            "sigv4": []
          }
        ],
        "x-amazon-apigateway-integration": {
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "responses": {
            "4\\d{2}": {
              "statusCode": "400"
            },
            "default": {
              "statusCode": "200"
            },
            "5\\d{2}": {
              "statusCode": "500"
            }
          },
          "requestParameters": {
            "integration.request.path.object": "method.request.path.item",
            "integration.request.path.bucket": "method.request.path.folder"
          },
          "uri": "arn:aws:apigateway:us-west-2:s3:path/{bucket}/{object}",
          "passthroughBehavior": "when_no_match",
          "httpMethod": "DELETE",
          "type": "aws"
        }
      }
```

```
    }
  },
  "securityDefinitions": {
    "sigv4": {
      "type": "apiKey",
      "name": "Authorization",
      "in": "header",
      "x-amazon-apigateway-authtype": "awsSigv4"
    }
  },
  "definitions": {
    "Empty": {
      "type": "object",
      "title": "Empty Schema"
    }
  }
}
```

OpenAPI 3.0

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "MyS3",
    "version" : "2016-10-13T23:04:43Z"
  },
  "servers" : [ {
    "url" : "https://9gn28ca086.execute-api.{region}.amazonaws.com/{basePath}",
    "variables" : {
      "basePath" : {
        "default" : "S3"
      }
    }
  } ],
  "paths" : {
    "/{folder}" : {
      "get" : {
        "parameters" : [ {
          "name" : "folder",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
```

```
          }
        } ],
        "responses" : {
          "400" : {
            "description" : "400 response",
            "content" : { }
          },
          "500" : {
            "description" : "500 response",
            "content" : { }
          },
          "200" : {
            "description" : "200 response",
            "headers" : {
              "Content-Length" : {
                "schema" : {
                  "type" : "string"
                }
              },
              "Date" : {
                "schema" : {
                  "type" : "string"
                }
              },
              "Content-Type" : {
                "schema" : {
                  "type" : "string"
                }
              }
            },
            "content" : {
              "application/json" : {
                "schema" : {
                  "$ref" : "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : {
          "credentials" : "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "httpMethod" : "GET",
          "uri" : "arn:aws:apigateway:us-west-2:s3:path/{bucket}",
          "responses" : {
```

```
            "4\\d{2}" : {
              "statusCode" : "400"
            },
            "default" : {
              "statusCode" : "200",
              "responseParameters" : {
                "method.response.header.Content-Type" :
"integration.response.header.Content-Type",
                "method.response.header.Date" : "integration.response.header.Date",
                "method.response.header.Content-Length" :
"integration.response.header.content-length"
              }
            },
            "5\\d{2}" : {
              "statusCode" : "500"
            }
          },
          "requestParameters" : {
            "integration.request.path.bucket" : "method.request.path.folder"
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "aws"
        }
      },
      "put" : {
        "parameters" : [ {
          "name" : "Content-Type",
          "in" : "header",
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "folder",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "400" : {
            "description" : "400 response",
            "content" : { }
          },
```

```
          "500" : {
            "description" : "500 response",
            "content" : { }
          },
          "200" : {
            "description" : "200 response",
            "headers" : {
              "Content-Length" : {
                "schema" : {
                  "type" : "string"
                }
              },
              "Content-Type" : {
                "schema" : {
                  "type" : "string"
                }
              }
            },
            "content" : {
              "application/json" : {
                "schema" : {
                  "$ref" : "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : {
          "credentials" : "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "httpMethod" : "PUT",
          "uri" : "arn:aws:apigateway:us-west-2:s3:path/{bucket}",
          "responses" : {
            "4\\d{2}" : {
              "statusCode" : "400"
            },
            "default" : {
              "statusCode" : "200",
              "responseParameters" : {
                "method.response.header.Content-Type" :
"integration.response.header.Content-Type",
                "method.response.header.Content-Length" :
"integration.response.header.Content-Length"
              }
            },
```

```
            "5\\d{2}" : {
              "statusCode" : "500"
            }
          },
          "requestParameters" : {
            "integration.request.path.bucket" : "method.request.path.folder",
            "integration.request.header.Content-Type" :
"method.request.header.Content-Type"
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "aws"
        }
      },
      "delete" : {
        "parameters" : [ {
          "name" : "folder",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "400" : {
            "description" : "400 response",
            "content" : { }
          },
          "500" : {
            "description" : "500 response",
            "content" : { }
          },
          "200" : {
            "description" : "200 response",
            "headers" : {
              "Date" : {
                "schema" : {
                  "type" : "string"
                }
              },
              "Content-Type" : {
                "schema" : {
                  "type" : "string"
                }
              }
```

```
            },
            "content" : {
              "application/json" : {
                "schema" : {
                  "$ref" : "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : {
          "credentials" : "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "httpMethod" : "DELETE",
          "uri" : "arn:aws:apigateway:us-west-2:s3:path/{bucket}",
          "responses" : {
            "4\\d{2}" : {
              "statusCode" : "400"
            },
            "default" : {
              "statusCode" : "200",
              "responseParameters" : {
                "method.response.header.Content-Type" :
 "integration.response.header.Content-Type",
                "method.response.header.Date" : "integration.response.header.Date"
              }
            },
            "5\\d{2}" : {
              "statusCode" : "500"
            }
          },
          "requestParameters" : {
            "integration.request.path.bucket" : "method.request.path.folder"
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "aws"
        }
      }
    },
    "/{folder}/{item}" : {
      "get" : {
        "parameters" : [ {
          "name" : "item",
          "in" : "path",
          "required" : true,
```

```
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "folder",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "400" : {
            "description" : "400 response",
            "content" : { }
          },
          "500" : {
            "description" : "500 response",
            "content" : { }
          },
          "200" : {
            "description" : "200 response",
            "headers" : {
              "content-type" : {
                "schema" : {
                  "type" : "string"
                }
              },
              "Content-Type" : {
                "schema" : {
                  "type" : "string"
                }
              }
            },
            "content" : {
              "application/json" : {
                "schema" : {
                  "$ref" : "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : {
```

```
          "credentials" : "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "httpMethod" : "GET",
          "uri" : "arn:aws:apigateway:us-west-2:s3:path/{bucket}/{object}",
          "responses" : {
            "4\\d{2}" : {
              "statusCode" : "400"
            },
            "default" : {
              "statusCode" : "200",
              "responseParameters" : {
                "method.response.header.content-type" :
"integration.response.header.content-type",
                "method.response.header.Content-Type" :
"integration.response.header.Content-Type"
              }
            },
            "5\\d{2}" : {
              "statusCode" : "500"
            }
          },
          "requestParameters" : {
            "integration.request.path.object" : "method.request.path.item",
            "integration.request.path.bucket" : "method.request.path.folder"
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "aws"
        }
      },
      "put" : {
        "parameters" : [ {
          "name" : "Content-Type",
          "in" : "header",
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "item",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "folder",
```

```
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "400" : {
            "description" : "400 response",
            "content" : { }
          },
          "500" : {
            "description" : "500 response",
            "content" : { }
          },
          "200" : {
            "description" : "200 response",
            "headers" : {
              "Content-Length" : {
                "schema" : {
                  "type" : "string"
                }
              },
              "Content-Type" : {
                "schema" : {
                  "type" : "string"
                }
              }
            },
            "content" : {
              "application/json" : {
                "schema" : {
                  "$ref" : "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : {
          "credentials" : "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "httpMethod" : "PUT",
          "uri" : "arn:aws:apigateway:us-west-2:s3:path/{bucket}/{object}",
          "responses" : {
            "4\\d{2}" : {
```

```
              "statusCode" : "400"
            },
            "default" : {
              "statusCode" : "200",
              "responseParameters" : {
                "method.response.header.Content-Type" :
"integration.response.header.Content-Type",
                "method.response.header.Content-Length" :
"integration.response.header.Content-Length"
              }
            },
            "5\\d{2}" : {
              "statusCode" : "500"
            }
          },
          "requestParameters" : {
            "integration.request.path.object" : "method.request.path.item",
            "integration.request.path.bucket" : "method.request.path.folder",
            "integration.request.header.Content-Type" :
"method.request.header.Content-Type"
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "aws"
        }
      },
      "delete" : {
        "parameters" : [ {
          "name" : "item",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "folder",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "400" : {
            "description" : "400 response",
```

```
            "content" : { }
          },
          "500" : {
            "description" : "500 response",
            "content" : { }
          },
          "200" : {
            "description" : "200 response",
            "headers" : {
              "Content-Length" : {
                "schema" : {
                  "type" : "string"
                }
              },
              "Content-Type" : {
                "schema" : {
                  "type" : "string"
                }
              }
            },
            "content" : {
              "application/json" : {
                "schema" : {
                  "$ref" : "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : {
          "credentials" : "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "httpMethod" : "DELETE",
          "uri" : "arn:aws:apigateway:us-west-2:s3:path/{bucket}/{object}",
          "responses" : {
            "4\\d{2}" : {
              "statusCode" : "400"
            },
            "default" : {
              "statusCode" : "200"
            },
            "5\\d{2}" : {
              "statusCode" : "500"
            }
          },
```

```
          "requestParameters" : {
            "integration.request.path.object" : "method.request.path.item",
            "integration.request.path.bucket" : "method.request.path.folder"
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "aws"
        }
      },
      "head" : {
        "parameters" : [ {
          "name" : "item",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "folder",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "400" : {
            "description" : "400 response",
            "content" : { }
          },
          "500" : {
            "description" : "500 response",
            "content" : { }
          },
          "200" : {
            "description" : "200 response",
            "headers" : {
              "Content-Length" : {
                "schema" : {
                  "type" : "string"
                }
              },
              "Content-Type" : {
                "schema" : {
                  "type" : "string"
```

```
                }
              }
            },
            "content" : {
              "application/json" : {
                "schema" : {
                  "$ref" : "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : {
          "credentials" : "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "httpMethod" : "HEAD",
          "uri" : "arn:aws:apigateway:us-west-2:s3:path/{bucket}/{object}",
          "responses" : {
            "4\\d{2}" : {
              "statusCode" : "400"
            },
            "default" : {
              "statusCode" : "200",
              "responseParameters" : {
                "method.response.header.Content-Type" :
"integration.response.header.Content-Type",
                "method.response.header.Content-Length" :
"integration.response.header.Content-Length"
              }
            },
            "5\\d{2}" : {
              "statusCode" : "500"
            }
          },
          "requestParameters" : {
            "integration.request.path.object" : "method.request.path.item",
            "integration.request.path.bucket" : "method.request.path.folder"
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "aws"
        }
      }
    },
    "/" : {
      "get" : {
```

```
        "responses" : {
          "400" : {
            "description" : "400 response",
            "content" : { }
          },
          "500" : {
            "description" : "500 response",
            "content" : { }
          },
          "200" : {
            "description" : "200 response",
            "headers" : {
              "Content-Length" : {
                "schema" : {
                  "type" : "string"
                }
              },
              "Timestamp" : {
                "schema" : {
                  "type" : "string"
                }
              },
              "Content-Type" : {
                "schema" : {
                  "type" : "string"
                }
              }
            },
            "content" : {
              "application/json" : {
                "schema" : {
                  "$ref" : "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : {
          "credentials" : "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "httpMethod" : "GET",
          "uri" : "arn:aws:apigateway:us-west-2:s3:path//",
          "responses" : {
            "4\\d{2}" : {
              "statusCode" : "400"
```

```
            },
            "default" : {
              "statusCode" : "200",
              "responseParameters" : {
                "method.response.header.Content-Type" :
 "integration.response.header.Content-Type",
                "method.response.header.Content-Length" :
 "integration.response.header.Content-Length",
                "method.response.header.Timestamp" :
 "integration.response.header.Date"
              }
            },
            "5\\d{2}" : {
              "statusCode" : "500"
            }
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "aws"
        }
      }
    }
  },
  "components" : {
    "schemas" : {
      "Empty" : {
        "title" : "Empty Schema",
        "type" : "object"
      }
    }
  }
}
```

Chiamalo API usando un client REST API

Per fornire un end-to-end tutorial, ora mostriamo come chiamare l'APIutilizzo di Postman, che supporta l' AWS IAMautorizzazione.

Per chiamare il nostro proxy Amazon S3 utilizzando Postman API

1. Implementa o ridistribuisci il. API Prendi nota URL della base API che viene visualizzata accanto a Invoke nella parte URL superiore dello Stage Editor.
2. Avvia Postman.

3. Seleziona Authorization (Autorizzazione), quindi scegli `AWS Signature`. Inserisci IAM l'ID della chiave di accesso e la chiave di accesso segreta dell'utente rispettivamente nei campi AccessKeye SecretKeydi immissione. Inserisci la Regione AWS destinazione in cui API viene distribuito nella casella di testo AWS Regione. Inserisci il nome del servizio **`execute-api`** nel campo di immissione.

   È possibile creare un paio di chiavi dalla scheda Credenziali di sicurezza del proprio account IAM utente nella Console di IAM gestione.

4. Per aggiungere un bucket denominato `amzn-s3-demo-bucket` all'account Amazon S3 nella regione *`{region}`*:

   a. Scegliete PUTdall'elenco a discesa dei metodi e digitate il metodo (URL`https://`*`api-id`*`.execute-api.`*`aws-region`*`.amazonaws.com/`*`stage`*`/`*`folder-name`*

   b. Imposta il valore dell'intestazione `Content-Type` come `application/xml`. Potrebbe essere necessario eliminare le intestazioni esistenti prima di impostare il tipo di contenuto.

   c. Scegliete la voce di menu Body e digitate il seguente XML frammento come corpo della richiesta:

   ```
   <CreateBucketConfiguration>
     <LocationConstraint>{region}</LocationConstraint>
   </CreateBucketConfiguration>
   ```

   d. Seleziona Send (Invia) per inviare la richiesta. Se l'operazione viene eseguita correttamente, dovresti ricevere una risposta `200 OK` con un payload vuoto.

5. Per aggiungere un file di testo a un bucket, segui le istruzioni riportate sopra. Se specificate il nome del bucket **`amzn-s3-demo-bucket`** for `{folder}` e il nome di file **`Readme.txt`** for `{item}` in URL e fornite una stringa di testo **`Hello, World!`** come contenuto del file (rendendolo quindi il payload della richiesta), la richiesta diventa

   ```
   PUT /S3/amzn-s3-demo-bucket/Readme.txt HTTP/1.1
   Host: 9gn28ca086.execute-api.{region}.amazonaws.com
   Content-Type: application/xml
   X-Amz-Date: 20161015T062647Z
   Authorization: AWS4-HMAC-SHA256 Credential=access-key-id/20161015/{region}/execute-
   api/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
    Signature=ccadb877bdb0d395ca38cc47e18a0d76bb5eaf17007d11e40bf6fb63d28c705b
   Cache-Control: no-cache
   Postman-Token: 6135d315-9cc4-8af8-1757-90871d00847e
   ```

```
Hello, World!
```

Se l'operazione viene eseguita correttamente, dovresti ricevere una risposta `200 OK` con un payload vuoto.

6. Per ottenere il contenuto del `Readme.txt` file che abbiamo appena aggiunto al `amzn-s3-demo-bucket` bucket, esegui una GET richiesta come la seguente:

```
GET /S3/amzn-s3-demo-bucket/Readme.txt HTTP/1.1
Host: 9gn28ca086.execute-api.{region}.amazonaws.com
Content-Type: application/xml
X-Amz-Date: 20161015T063759Z
Authorization: AWS4-HMAC-SHA256 Credential=access-key-id/20161015/{region}/
execute-api/aws4_request, SignedHeaders=content-type;host;x-amz-date,
 Signature=ba09b72b585acf0e578e6ad02555c00e24b420b59025bc7bb8d3f7aed1471339
Cache-Control: no-cache
Postman-Token: d60fcb59-d335-52f7-0025-5bd96928098a
```

Se l'operazione viene eseguita correttamente, dovresti ricevere una risposta `200 OK` con la stringa di testo `Hello, World!` come payload.

7. Per elencare le voci nel bucket `amzn-s3-demo-bucket`, invia la richiesta seguente:

```
GET /S3/amzn-s3-demo-bucket HTTP/1.1
Host: 9gn28ca086.execute-api.{region}.amazonaws.com
Content-Type: application/xml
X-Amz-Date: 20161015T064324Z
Authorization: AWS4-HMAC-SHA256 Credential=access-key-id/20161015/{region}/
execute-api/aws4_request, SignedHeaders=content-type;host;x-amz-date,
 Signature=4ac9bd4574a14e01568134fd16814534d9951649d3a22b3b0db9f1f5cd4dd0ac
Cache-Control: no-cache
Postman-Token: 9c43020a-966f-61e1-81af-4c49ad8d1392
```

In caso di successo, dovresti ricevere una `200 OK` risposta con un XML payload che mostri un singolo elemento nel bucket specificato, a meno che tu non abbia aggiunto altri file al bucket prima di inviare questa richiesta.

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Name>apig-demo-5</Name>
    <Prefix></Prefix>
    <Marker></Marker>
```

```
    <MaxKeys>1000</MaxKeys>
    <IsTruncated>false</IsTruncated>
    <Contents>
        <Key>Readme.txt</Key>
        <LastModified>2016-10-15T06:26:48.000Z</LastModified>
        <ETag>"65a8e27d8879283831b664bd8b7f0ad4"</ETag>
        <Size>13</Size>
        <Owner>
            <ID>06e4b09e9d...603addd12ee</ID>
            <DisplayName>user-name</DisplayName>
        </Owner>
        <StorageClass>STANDARD</StorageClass>
    </Contents>
</ListBucketResult>
```

> **Note**
>
> Per caricare o scaricare un'immagine, devi impostare la gestione dei contenuti su _TO_. CONVERT BINARY

Tutorial: creazione di un'API REST come un proxy Amazon Kinesis

In questa pagina viene descritto come creare e configurare un'API REST con un'integrazione di tipo `AWS` per accedere a Kinesis.

> **Note**
>
> Per integrare l'API di API Gateway con Kinesis, devi scegliere una regione in cui i servizi API Gateway e Kinesis siano entrambi disponibili. Per la disponibilità della regione, vedere Endpoint e quote del servizio.

Per scopi illustrativi, creeremo un'API di esempio per permettere a un client di eseguire le operazioni seguenti:

1. Elencare i flussi disponibili dell'utente in Kinesis
2. Creare, descrivere o eliminare un flusso specificato.
3. Leggere o scrivere record di dati nel flusso specificato

Per eseguire le attività precedenti, l'API espone metodi in diverse risorse per richiamare, rispettivamente, gli elementi seguenti:

1. L'operazione `ListStreams` in Kinesis
2. Operazione `CreateStream`, `DescribeStream` o `DeleteStream`
3. Operazione `GetRecords` o `PutRecords` (inclusa `PutRecord`) in Kinesis

In particolare, compileremo l'API in questo modo:

- Esponi un metodo HTTP GET sulla `/streams` risorsa dell'API e integra il metodo con l'ListStreamsazione in Kinesis per elencare gli stream nell'account del chiamante.
- Esponi un metodo HTTP POST sulla `/streams/{stream-name}` risorsa dell'API e integra il metodo con l'CreateStreamazione in Kinesis per creare uno stream denominato nell'account del chiamante.
- Esponi un metodo HTTP GET sulla `/streams/{stream-name}` risorsa dell'API e integra il metodo con l'DescribeStreamazione in Kinesis per descrivere uno stream denominato nell'account del chiamante.
- Esponi un metodo HTTP DELETE sulla `/streams/{stream-name}` risorsa dell'API e integra il metodo con l'DeleteStreamazione in Kinesis per eliminare uno stream nell'account del chiamante.
- Esponi un metodo HTTP PUT sulla `/streams/{stream-name}/record` risorsa dell'API e integra il metodo con l'PutRecordazione in Kinesis. In questo modo, il client può aggiungere un singolo record di dati al flusso denominato.
- Esponi un metodo HTTP PUT sulla `/streams/{stream-name}/records` risorsa dell'API e integra il metodo con l'PutRecordsazione in Kinesis. In questo modo, il client può aggiungere un elenco di record di dati al flusso denominato.
- Esponi un metodo HTTP GET sulla `/streams/{stream-name}/records` risorsa dell'API e integra il metodo con l'GetRecordsazione in Kinesis. In questo modo, il client può aggiungere un elenco di record di dati nel flusso denominato, con un'iterazione shard specificata. Un'iterazione shard specifica la posizione dello shard da cui iniziare a leggere i record di dati in sequenza.
- Esponi un metodo HTTP GET sulla `/streams/{stream-name}/sharditerator` risorsa dell'API e integra il metodo con l'GetShardIteratorazione in Kinesis. Questo metodo helper deve essere fornito all'operazione `ListStreams` in Kinesis.

Puoi applicare le istruzioni presentate qui ad altre operazioni di Kinesis. Per l'elenco completo delle operazioni di Kinesis, consulta la documentazione di riferimento delle API di Amazon Kinesis.

Invece di usare la console API Gateway per creare l'API di esempio, puoi importare l'API di esempio in API Gateway, usando l'API Gateway di Importa API. Per informazioni su come usare l'API di importazione, consulta Sviluppa REST APIs utilizzando Open API in API Gateway.

Creazione di un ruolo e una policy IAM per l'API per accedere a Kinesis

Per permettere all'API di richiamare le operazioni di Kinesis, devi aver collegato le policy IAM appropriate a un ruolo IAM.

Per creare il ruolo di esecuzione del proxy del AWS servizio

1. Accedi AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Scegliere Roles (Ruoli).
3. Scegli Crea ruolo.
4. Scegli il AWS servizio in Seleziona il tipo di entità affidabile, quindi seleziona API Gateway e seleziona Consenti a API Gateway di inviare i log ai CloudWatch log.
5. Scegli Successivo e di nuovo Successivo.
6. In Role name (Nome ruolo) immettere **`APIGatewayKinesisProxyPolicy`** e quindi selezionare Create role (Crea ruolo).
7. Nell'elenco Roles (Ruoli) scegliere il ruolo appena creato. Potrebbe essere necessario scorrere o utilizzare la barra di ricerca per trovare il ruolo.
8. Per il ruolo selezionato, seleziona la scheda Aggiungi autorizzazioni.
9. Dall'elenco a discesa scegli Collega policy.
10. Nella barra di ricerca inserisci **`AmazonKinesisFullAccess`** e scegli Aggiungi autorizzazioni.

    > **Note**
    >
    > Per semplicità, questo tutorial utilizza una policy gestita. Come best practice, dovresti creare la tua policy IAM per concedere le autorizzazioni minime richieste.

11. Annota l'ARN del ruolo appena creato, lo utilizzerai in seguito.

Crea un'API come proxy Kinesis

Usa la procedura seguente per creare l'API nella console API Gateway.

Per creare un'API come proxy AWS di servizio per Kinesis

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Se si utilizza API Gateway per la prima volta, verrà visualizzata una pagina che presenta le caratteristiche del servizio. In API REST, scegliere Crea. Quando appare il popup Create Example API (Crea API di esempio), scegliere OK.

   Se non è la prima volta che si utilizza API Gateway, scegliere Create API (Crea API). In API REST, scegliere Crea.
3. Selezionare New API (Nuova API).
4. Per API name (Nome API) immettere **`KinesisProxy`**. Per tutti gli altri campi mantieni i valori predefiniti.
5. (Facoltativo) In Description (Descrizione), immettere una descrizione.
6. Seleziona Create API (Crea API).

Dopo la creazione dell'API, la console API Gateway visualizza la pagina Resources (Risorse), che contiene solo la risorsa root (`/`) dell'API.

Visualizzazione dell'elenco di flussi in Kinesis

Kinesis supporta l'operazione `ListStreams` con la chiamata API REST seguente:

```
POST /?Action=ListStreams HTTP/1.1
Host: kinesis.<region>.<domain>
Content-Length: <PayloadSizeBytes>
User-Agent: <UserAgentString>
Content-Type: application/x-amz-json-1.1
Authorization: <AuthParams>
X-Amz-Date: <Date>

{
   ...
}
```

Nella richiesta API REST precedente l'operazione è specificata nel parametro di query `Action`. In alternativa, puoi specificare l'operazione in un'intestazione `X-Amz-Target`:

```
POST / HTTP/1.1
Host: kinesis.<region>.<domain>
```

```
Content-Length: <PayloadSizeBytes>
User-Agent: <UserAgentString>
Content-Type: application/x-amz-json-1.1
Authorization: <AuthParams>
X-Amz-Date: <Date>
X-Amz-Target: Kinesis_20131202.ListStreams
{
   ...
}
```

In questo tutorial useremo il parametro di query per specificare l'operazione.

Per esporre un'operazione di Kinesis nell'API, aggiungi una risorsa `/streams` alla root dell'API. Imposta quindi un metodo `GET` nella risorsa e integra il metodo con l'operazione `ListStreams` di Kinesis.

La procedura seguente descrive come elencare flussi Kinesis tramite la console API Gateway.

Per elencare i flussi Kinesis utilizzando la console API Gateway

1. Seleziona la risorsa `/`, quindi scegli Crea risorsa.
2. Per Resource Name (Nome risorsa) immetti **`streams`**.
3. Mantieni CORS (Cross Origin Resource Sharing) disattivato.
4. Scegli Crea risorsa.
5. Scegli la risorsa `/streams` e seleziona Crea metodo, quindi procedi come segue:

   a. Per Tipo di metodo seleziona GET.

   > **Note**
   >
   > Il verbo HTTP per un metodo richiamato da un client può differire dal verbo HTTP per un'integrazione richiesta dal back-end. Qui selezioniamo `GET` perché la visualizzazione dell'elenco di flussi è intuitivamente un'operazione READ.

   b. Per Tipo di integrazione seleziona Servizio AWS .
   c. Per Regione AWS, seleziona il Regione AWS luogo in cui hai creato lo stream Kinesis.
   d. Per Servizio AWS seleziona Kinesis.
   e. Lascia vuoto Sottodominio AWS .
   f. Per HTTP method (Metodo HTTP) scegli POST.

> **Note**
>
> Qui scegliamo `POST` perché Kinesis richiede che con il metodo venga richiamata l'operazione `ListStreams`.

g. Per Tipo di operazione scegli Usa nome operazione.

h. Per Nome azione immetti **`ListStreams`**.

i. Per Ruolo di esecuzione immetti l'ARN del ruolo di esecuzione.

j. Lascia il valore predefinito Transito per Gestione contenuti.

k. Scegli Crea metodo.

6. Nella scheda Richiesta di integrazione scegli Modifica in Impostazioni della richiesta di integrazione.
7. Per Richiesta corpo passthrough scegli Quando non ci sono modelli definiti (consigliato).
8. Scegli Parametri delle intestazioni delle richieste URL ed effettua le seguenti operazioni:

   a. Scegli Aggiungi parametro delle intestazioni delle richieste.

   b. Per Nome, immetti **`Content-Type`**.

   c. In Mappato da, inserire **`'application/x-amz-json-1.1'`**.

   Usiamo la mappatura dei parametri delle richieste per impostare l'intestazione `Content-Type` sul valore statico `'application/x-amz-json-1.1'` per indicare a Kinesis che l'input è una versione specifica di JSON.

9. Scegli Modelli di mappatura, quindi seleziona Aggiungi modello di mappatura ed effettua le seguenti operazioni:

   a. Per Content-Type immetti **`application/json`**.

   b. Per Corpo del modello immetti **`{}`**.

   c. Selezionare Salva.

   La ListStreamsrichiesta richiede un payload nel seguente formato JSON:

```
{
    "ExclusiveStartStreamName": "string",
```

```
    "Limit": number
}
```

Tuttavia, le proprietà sono facoltative. Per usare i valori predefiniti, qui abbiamo optato per un payload JSON vuoto.

10. Testa il metodo GET nella risorsa /streams per richiamare l'azione `ListStreams` in Kinesis:

    Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.

    Scegli Esegui test per testare il metodo.

    Se hai già creato due flussi denominati "myStream" e "yourStream" in Kinesis, il test con esito positivo restituisce una risposta 200 OK che contiene il payload seguente:

```
{
     "HasMoreStreams": false,
     "StreamNames": [
         "myStream",
         "yourStream"
     ]
}
```

Creazione, descrizione ed eliminazione di un flusso in Kinesis

Per creare, descrivere ed eliminare un flusso in Kinesis è necessario effettuare, rispettivamente, le richieste API REST di Kinesis seguenti:

```
POST /?Action=CreateStream HTTP/1.1
Host: kinesis.region.domain
...
Content-Type: application/x-amz-json-1.1
Content-Length: PayloadSizeBytes

{
    "ShardCount": number,
```

```
    "StreamName": "string"
}
```

```
POST /?Action=DescribeStream HTTP/1.1
Host: kinesis.region.domain
...
Content-Type: application/x-amz-json-1.1
Content-Length: PayloadSizeBytes

{
    "StreamName": "string"
}
```

```
POST /?Action=DeleteStream HTTP/1.1
Host: kinesis.region.domain
...
Content-Type: application/x-amz-json-1.1
Content-Length: PayloadSizeBytes

{
    "StreamName":"string"
}
```

Puoi compilare l'API in modo da accettare l'input richiesto come payload JSON della richiesta del metodo e passare il payload alla richiesta di integrazione. Tuttavia, per fornire più esempi di mappatura dei dati tra richieste di metodi e di integrazione e le rispettive risposte, creeremo l'API in un modo diverso.

Esponiamo i metodi `GETPOST`, e `Delete` HTTP su una risorsa. to-be-named `Stream` Useremo la variabile di percorso `{stream-name}` come segnaposto della risorsa flusso e integreremo questi metodi API rispettivamente con le operazioni `DescribeStream`, `CreateStream` e `DeleteStream` di Kinesis. Richiederemo che il client passi altri dati di input come intestazioni, parametri di query o payload della richiesta di un metodo. Specificheremo modelli di mappatura per trasformare i dati nel payload della richiesta di integrazione necessario.

Per creare la risorsa {stream-name}

1. Scegli la risorsa /streams, quindi scegli Crea risorsa.
2. Mantieni l'opzione Risorsa proxy disattivata.
3. Per Percorso risorsa seleziona `/streams`.
4. Per Resource Name (Nome risorsa) immetti **`{stream-name}`**.
5. Mantieni CORS (Cross Origin Resource Sharing) disattivato.
6. Scegli Crea risorsa.

Per configurare e testare il metodo GET in una risorsa flusso

1. Scegli la risorsa/{stream-name}, quindi scegli il metodo Create.
2. Per Tipo di metodo seleziona GET.
3. Per Tipo di integrazione seleziona Servizio AWS .
4. Per Regione AWS, seleziona il Regione AWS luogo in cui hai creato lo stream Kinesis.
5. Per Servizio AWS seleziona Kinesis.
6. Lascia vuoto Sottodominio AWS .
7. Per HTTP method (Metodo HTTP) scegli POST.
8. Per Tipo di operazione scegli Usa nome operazione.
9. Per Nome azione immetti **`DescribeStream`**.
10. Per Ruolo di esecuzione immetti l'ARN del ruolo di esecuzione.
11. Lascia il valore predefinito Transito per Gestione contenuti.
12. Scegli Crea metodo.
13. Nella sezione Richiesta di integrazione aggiungi i seguenti parametri delle intestazioni delle richieste URL:

    ```
    Content-Type: 'x-amz-json-1.1'
    ```

    L'attività segue la stessa procedura usata per configurare la mappatura dei parametri di richiesta per il metodo `GET /streams`.
14. Aggiungi il modello di mappatura del corpo seguente per mappare i dati dalla richiesta del metodo `GET /streams/{stream-name}` alla richiesta di integrazione `POST /?Action=DescribeStream`:

```
{
    "StreamName": "$input.params('stream-name')"
}
```

Questo modello di mappatura genera il payload della richiesta di integrazione necessario per l'operazione `DescribeStream` di Kinesis dal valore del parametro di percorso `stream-name` della richiesta del metodo.

15. Per testare il metodo `GET /stream/{stream-name}` per richiamare l'azione `DescribeStream` in Kinesis, scegli la scheda Test.
16. Per Percorso immetti il nome di un flusso Kinesis esistente in stream-name.
17. Scegli Test (Esegui test). Se il test ha esito positivo, viene restituita una risposta 200 OK con un payload simile al seguente:

```
{
  "StreamDescription": {
    "HasMoreShards": false,
    "RetentionPeriodHours": 24,
    "Shards": [
      {
        "HashKeyRange": {
          "EndingHashKey": "68056473384187692692674921486353642290",
          "StartingHashKey": "0"
        },
        "SequenceNumberRange": {
          "StartingSequenceNumber":
 "49559266461454070523309915164834022007924120923395850242"
        },
        "ShardId": "shardId-000000000000"
      },
      ...
      {
        "HashKeyRange": {
          "EndingHashKey": "340282366920938463463374607431768211455",
          "StartingHashKey": "272225893536750770770699685945414569164"
        },
        "SequenceNumberRange": {
          "StartingSequenceNumber":
 "49559266461543273504104037657400164881014714369419771970"
        },
        "ShardId": "shardId-000000000004"
```

```
      }
    ],
    "StreamARN": "arn:aws:kinesis:us-east-1:12345678901:stream/myStream",
    "StreamName": "myStream",
    "StreamStatus": "ACTIVE"
  }
}
```

Dopo aver distribuito l'API, puoi effettuare una richiesta REST su questo metodo API:

```
GET https://your-api-id.execute-api.region.amazonaws.com/stage/streams/myStream
 HTTP/1.1
Host: your-api-id.execute-api.region.amazonaws.com
Content-Type: application/json
Authorization: ...
X-Amz-Date: 20160323T194451Z
```

Per configurare e testare il metodo POST in una risorsa flusso

1. Scegli la risorsa/{stream-name}, quindi scegli il metodo Create.
2. Per Tipo di metodo seleziona POST.
3. Per Tipo di integrazione seleziona Servizio AWS .
4. Per Regione AWS, seleziona il Regione AWS luogo in cui hai creato lo stream Kinesis.
5. Per Servizio AWS seleziona Kinesis.
6. Lascia vuoto Sottodominio AWS .
7. Per HTTP method (Metodo HTTP) scegli POST.
8. Per Tipo di operazione scegli Usa nome operazione.
9. Per Nome azione immetti **`CreateStream`**.
10. Per Ruolo di esecuzione immetti l'ARN del ruolo di esecuzione.
11. Lascia il valore predefinito Transito per Gestione contenuti.
12. Scegli Crea metodo.
13. Nella sezione Richiesta di integrazione aggiungi i seguenti parametri delle intestazioni delle richieste URL:

```
Content-Type: 'x-amz-json-1.1'
```

L'attività segue la stessa procedura usata per configurare la mappatura dei parametri di richiesta per il metodo `GET /streams`.

14. Aggiungi il modello di mappatura del corpo seguente per mappare i dati dalla richiesta del metodo `POST /streams/{stream-name}` alla richiesta di integrazione `POST /?Action=CreateStream`:

```
{
    "ShardCount": #if($input.path('$.ShardCount') == '') 5 #else
 $input.path('$.ShardCount') #end,
    "StreamName": "$input.params('stream-name')"
}
```

Nel modello di mappatura precedente impostiamo `ShardCount` sul valore fisso 5 se il client non specifica alcun valore nel payload della richiesta del metodo.

15. Per testare il metodo `POST /stream/{stream-name}` per richiamare l'azione `CreateStream` in Kinesis, scegli la scheda Test.
16. Per Percorso immetti il nome di un nuovo flusso Kinesis in stream-name.
17. Scegli Test (Esegui test). Se il test ha esito positivo, viene restituita una risposta 200 OK senza dati.

Dopo aver distribuito l'API, puoi anche effettuare una richiesta API REST sul metodo POST in una risorsa flusso per richiamare l'operazione `CreateStream` in Kinesis:

```
POST https://your-api-id.execute-api.region.amazonaws.com/stage/streams/yourStream
 HTTP/1.1
Host: your-api-id.execute-api.region.amazonaws.com
Content-Type: application/json
Authorization: ...
X-Amz-Date: 20160323T194451Z

{
    "ShardCount": 5
}
```

Configurazione e test del metodo DELETE in una risorsa flusso

1. Scegli la risorsa/{stream-name}, quindi scegli il metodo Create.
2. Per Tipo di metodo seleziona DELETE.
3. Per Tipo di integrazione seleziona Servizio AWS .
4. Per Regione AWS, seleziona il Regione AWS luogo in cui hai creato lo stream Kinesis.
5. Per Servizio AWS seleziona Kinesis.
6. Lascia vuoto Sottodominio AWS .
7. Per HTTP method (Metodo HTTP) scegli POST.
8. Per Tipo di operazione scegli Usa nome operazione.
9. Per Nome azione immetti **`DeleteStream`**.
10. Per Ruolo di esecuzione immetti l'ARN del ruolo di esecuzione.
11. Lascia il valore predefinito Transito per Gestione contenuti.
12. Scegli Crea metodo.
13. Nella sezione Richiesta di integrazione aggiungi i seguenti parametri delle intestazioni delle richieste URL:

    ```
    Content-Type: 'x-amz-json-1.1'
    ```

    L'attività segue la stessa procedura usata per configurare la mappatura dei parametri di richiesta per il metodo `GET /streams`.
14. Aggiungi il modello di mappatura del corpo seguente per mappare i dati dalla richiesta del metodo `DELETE /streams/{stream-name}` alla richiesta di integrazione corrispondente di `POST /?Action=DeleteStream`:

    ```
    {
        "StreamName": "$input.params('stream-name')"
    }
    ```

    Questo modello di mappatura genera l'input richiesto per l'operazione `DELETE /streams/{stream-name}` dal nome di percorso URL fornito dal client `stream-name`.
15. Per testare il metodo `DELETE /stream/{stream-name}` per richiamare l'azione `DeleteStream` in Kinesis, scegli la scheda Test.
16. Per Percorso immetti il nome di un flusso Kinesis esistente in stream-name.

17. Scegli Test (Esegui test). Se il test ha esito positivo, viene restituita una risposta 200 OK senza dati.

    Dopo aver distribuito l'API, puoi anche effettuare la richiesta API REST seguente sul metodo DELETE nella risorsa flusso per chiamare l'operazione `DeleteStream` in Kinesis:

```
DELETE https://your-api-id.execute-api.region.amazonaws.com/stage/
streams/yourStream HTTP/1.1
Host: your-api-id.execute-api.region.amazonaws.com
Content-Type: application/json
Authorization: ...
X-Amz-Date: 20160323T194451Z

{}
```

Recupero di record e aggiunta di record in un flusso in Kinesis

Dopo aver creato un flusso in Kinesis, puoi aggiungere record di dati al flusso e leggere i dati dal flusso. L'aggiunta di record di dati implica la chiamata all'PutRecordazione PutRecordso in Kinesis. La prima operazione aggiunge più record, la seconda aggiunge un singolo record al flusso.

```
POST /?Action=PutRecords HTTP/1.1
Host: kinesis.region.domain
Authorization: AWS4-HMAC-SHA256 Credential=..., ...
...
Content-Type: application/x-amz-json-1.1
Content-Length: PayloadSizeBytes

{
    "Records": [
        {
            "Data": blob,
            "ExplicitHashKey": "string",
            "PartitionKey": "string"
        }
    ],
    "StreamName": "string"
}
```

oppure

```
POST /?Action=PutRecord HTTP/1.1
Host: kinesis.region.domain
Authorization: AWS4-HMAC-SHA256 Credential=..., ...
...
Content-Type: application/x-amz-json-1.1
Content-Length: PayloadSizeBytes

{
    "Data": blob,
    "ExplicitHashKey": "string",
    "PartitionKey": "string",
    "SequenceNumberForOrdering": "string",
    "StreamName": "string"
}
```

Qui `StreamName` identifica il flusso di destinazione per aggiungere record. `StreamName`, `Data`, e `PartitionKey` sono dati di input obbligatori. Nel nostro esempio useremo i valori predefiniti per tutti i dati di input facoltativi e non ne specificheremo in modo esplicito i valori nell'input per la richiesta del metodo.

Leggere i dati in Kinesis equivale a richiamare l'azione GetRecords:

```
POST /?Action=GetRecords HTTP/1.1
Host: kinesis.region.domain
Authorization: AWS4-HMAC-SHA256 Credential=..., ...
...
Content-Type: application/x-amz-json-1.1
Content-Length: PayloadSizeBytes

{
    "ShardIterator": "string",
    "Limit": number
}
```

Qui il flusso di origine da cui vogliamo ottenere record è specificato nel valore `ShardIterator` obbligatorio, come mostrato nell'operazione di Kinesis seguente per ottenere un'iterazione shard:

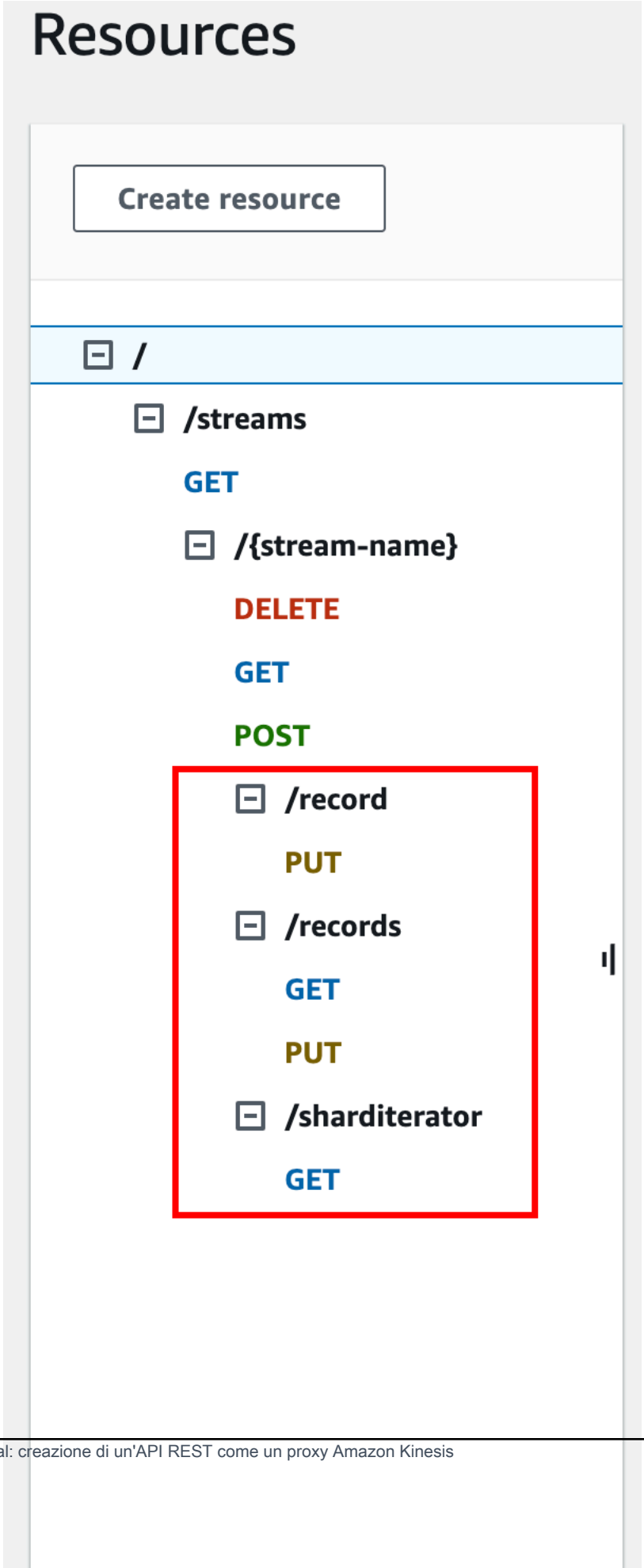
```
POST /?Action=GetShardIterator HTTP/1.1
Host: kinesis.region.domain
Authorization: AWS4-HMAC-SHA256 Credential=..., ...
...
Content-Type: application/x-amz-json-1.1
Content-Length: PayloadSizeBytes

{
    "ShardId": "string",
    "ShardIteratorType": "string",
    "StartingSequenceNumber": "string",
    "StreamName": "string"
}
```

Per le operazioni `GetRecords` e `PutRecords`, esponiamo rispettivamente i metodi `GET` e PUT in una risorsa `/records` che viene aggiunta a una risorsa flusso denominata (`/{stream-name}`). Analogamente, esponiamo l'operazione `PutRecord` come metodo PUT in una risorsa `/record`.

Poiché l'operazione `GetRecords` accetta come input un valore `ShardIterator`, ottenuto chiamando l'operazione helper `GetShardIterator`, esponiamo un metodo helper `GET` in una risorsa `ShardIterator` (`/sharditerator`).

Per creare le risorse /record, /records e /sharditerator

1. Scegli la risorsa/{stream-name}, quindi scegli Crea risorsa.
2. Mantieni l'opzione Risorsa proxy disattivata.
3. Per Percorso risorsa seleziona `/{stream-name}`.
4. Per Resource Name (Nome risorsa) immetti **`record`**.
5. Mantieni CORS (Cross Origin Resource Sharing) disattivato.
6. Scegli Crea risorsa.
7. Ripeti le fasi precedenti per creare una risorsa /records e una risorsa /sharditerator. L'API finale sarà simile alla seguente:


Resources
Create resource
/
/streams
GET
/{stream-name}
DELETE
GET
POST
/record
PUT
/records
GET
PUT
/sharditerator
GET

Le quattro procedure seguenti descrivono come configurare ognuno dei metodi, come mappare i dati dalle richieste dei metodi alle richieste di integrazione e come testare i metodi.

Per configurare e testare il metodo **`PUT /streams/{stream-name}/record`** per richiamare **`PutRecord`** in Kinesis:

1. Scegli /record, quindi scegli il metodo Create.
2. Per Tipo di metodo seleziona PUT.
3. Per Tipo di integrazione seleziona Servizio AWS .
4. Per Regione AWS, seleziona il Regione AWS luogo in cui hai creato lo stream Kinesis.
5. Per Servizio AWS seleziona Kinesis.
6. Lascia vuoto Sottodominio AWS .
7. Per HTTP method (Metodo HTTP) scegli POST.
8. Per Tipo di operazione scegli Usa nome operazione.
9. Per Nome azione immetti **`PutRecord`**.
10. Per Ruolo di esecuzione immetti l'ARN del ruolo di esecuzione.
11. Lascia il valore predefinito Transito per Gestione contenuti.
12. Scegli Crea metodo.
13. Nella sezione Richiesta di integrazione aggiungi i seguenti parametri delle intestazioni delle richieste URL:

    ```
    Content-Type: 'x-amz-json-1.1'
    ```

    L'attività segue la stessa procedura usata per configurare la mappatura dei parametri di richiesta per il metodo `GET /streams`.

14. Aggiungi il modello di mappatura del corpo seguente per mappare i dati dalla richiesta del metodo `PUT /streams/{stream-name}/record` alla richiesta di integrazione corrispondente di `POST /?Action=PutRecord`:

    ```
    {
        "StreamName": "$input.params('stream-name')",
        "Data": "$util.base64Encode($input.json('$.Data'))",
        "PartitionKey": "$input.path('$.PartitionKey')"
    }
    ```

Questo modello di mappatura presuppone che il payload della richiesta del metodo abbia il formato seguente:

```
{
   "Data": "some data",
   "PartitionKey": "some key"
}
```

Questi dati possono essere modellati tramite lo schema JSON seguente:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "PutRecord proxy single-record payload",
  "type": "object",
  "properties": {
      "Data": { "type": "string" },
      "PartitionKey": { "type": "string" }
  }
}
```

Puoi creare un modello per includere questo schema e usare il modello per semplificare la generazione del modello di mappatura. Tuttavia, puoi generare un modello di mappatura senza usare alcun modello.

15. Per testare il metodo `PUT /streams/{stream-name}/record`, imposta la variabile di percorso `stream-name` sul nome di un flusso esistente, fornisci un payload con il formato richiesto e quindi invia la richiesta del metodo. Il risultato con esito positivo è una risposta `200 OK` con un payload nel formato seguente:

```
{
  "SequenceNumber": "49559409944537880850133345460169886593573102115167928386",
  "ShardId": "shardId-000000000004"
}
```

Per configurare e testare il metodo **`PUT /streams/{stream-name}/records`** per richiamare **`PutRecords`** in Kinesis

1. Scegli la risorsa /records, quindi scegli il metodo Create.
2. Per Tipo di metodo seleziona PUT.
3. Per Tipo di integrazione seleziona Servizio AWS .
4. Per Regione AWS, seleziona il Regione AWS luogo in cui hai creato lo stream Kinesis.
5. Per Servizio AWS seleziona Kinesis.
6. Lascia vuoto Sottodominio AWS .
7. Per HTTP method (Metodo HTTP) scegli POST.
8. Per Tipo di operazione scegli Usa nome operazione.
9. Per Nome azione immetti **`PutRecords`**.
10. Per Ruolo di esecuzione immetti l'ARN del ruolo di esecuzione.
11. Lascia il valore predefinito Transito per Gestione contenuti.
12. Scegli Crea metodo.
13. Nella sezione Richiesta di integrazione aggiungi i seguenti parametri delle intestazioni delle richieste URL:

    ```
    Content-Type: 'x-amz-json-1.1'
    ```

    L'attività segue la stessa procedura usata per configurare la mappatura dei parametri di richiesta per il metodo `GET /streams`.
14. Aggiungi il modello di mappatura seguente per mappare i dati dalla richiesta del metodo `PUT /streams/{stream-name}/records` alla richiesta di integrazione corrispondente di `POST /?Action=PutRecords`:

    ```
    {
        "StreamName": "$input.params('stream-name')",
        "Records": [
           #foreach($elem in $input.path('$.records'))
              {
                "Data": "$util.base64Encode($elem.data)",
                "PartitionKey": "$elem.partition-key"
              }#if($foreach.hasNext),#end
            #end
        ]
    ```

```
}
```

Questo modello di mappatura presuppone che il payload della richiesta del metodo possa essere modellato tramite lo schema JSON seguente:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "PutRecords proxy payload data",
  "type": "object",
  "properties": {
    "records": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "data": { "type": "string" },
          "partition-key": { "type": "string" }
        }
      }
    }
  }
}
```

Puoi creare un modello per includere questo schema e usare il modello per semplificare la generazione del modello di mappatura. Tuttavia, puoi generare un modello di mappatura senza usare alcun modello.

In questo tutorial abbiamo usato due formati di payload leggermente diversi per mostrare come uno sviluppatore di API possa scegliere di esporre il formato di dati di back-end al client o nasconderlo dal client. Un formato è per il metodo PUT `/streams/{stream-name}/records` (sopra). Un altro formato viene usato per il metodo PUT `/streams/{stream-name}/record` (nella procedura precedente). Nell'ambiente di produzione devi mantenere i due formati coerenti.

15. Per testare il metodo PUT `/streams/{stream-name}/records`, imposta la variabile di percorso `stream-name` su un flusso esistente, fornisci il payload seguente e invia la richiesta del metodo.

```
{
    "records": [
```

```
        {
            "data": "some data",
            "partition-key": "some key"
        },
        {
            "data": "some other data",
            "partition-key": "some key"
        }
    ]
}
```

Il risultato con esito positivo è una risposta 200 OK con un payload simile all'output seguente:

```
{
  "FailedRecordCount": 0,
  "Records": [
    {
      "SequenceNumber": "49559409944537880850133345460167468741933742152373764162",
      "ShardId": "shardId-000000000004"
    },
    {
      "SequenceNumber": "49559409944537880850133345460168677667753356781548470338",
      "ShardId": "shardId-000000000004"
    }
  ]
}
```

Per configurare e testare il metodo **`GET /streams/{stream-name}/sharditerator`** per richiamare **`GetShardIterator`** in Kinesis

Il metodo `GET /streams/{stream-name}/sharditerator` è un metodo helper per acquisire un'iterazione shard necessaria prima di chiamare il metodo `GET /streams/{stream-name}/records`.

1. Scegli la risorsa /sharditerator, quindi scegli il metodo Create.
2. Per Tipo di metodo seleziona GET.
3. Per Tipo di integrazione seleziona Servizio AWS .
4. Per Regione AWS, seleziona il Regione AWS luogo in cui hai creato lo stream Kinesis.

5. Per Servizio AWS seleziona Kinesis.
6. Lascia vuoto Sottodominio AWS .
7. Per HTTP method (Metodo HTTP) scegli POST.
8. Per Tipo di operazione scegli Usa nome operazione.
9. Per Nome azione immetti **`GetShardIterator`**.
10. Per Ruolo di esecuzione immetti l'ARN del ruolo di esecuzione.
11. Lascia il valore predefinito Transito per Gestione contenuti.
12. Scegli Parametri della stringa di query URL.

    L'`GetShardIterator`azione richiede l'immissione di un ShardId valore. Per passare un valore `ShardId` fornito dal client, aggiungiamo un parametro di query `shard-id` alla richiesta del metodo, come mostrato nella seguente fase.
13. Scegliere Add query string (Aggiungi stringa di query).
14. Per Nome, immetti **`shard-id`**.
15. Mantieni Obbligatorio e Caching disattivati.
16. Scegli Crea metodo.
17. Nella sezione Richiesta di integrazione aggiungi il seguente modello di mappatura per generare l'input richiesto (`ShardId` e `StreamName`) per l'azione `GetShardIterator` dei parametri `shard-id` e `stream-name` della richiesta del metodo. Inoltre, il modello di mappatura imposta anche `ShardIteratorType` su `TRIM_HORIZON` come comportamento predefinito.

    ```
    {
        "ShardId": "$input.params('shard-id')",
        "ShardIteratorType": "TRIM_HORIZON",
        "StreamName": "$input.params('stream-name')"
    }
    ```
18. Usando l'opzione Test nella console API Gateway, immetti il nome di un flusso esistente come valore della variabile `stream-name`Path (Percorso), imposta `shard-id`Query string (Stringa di query) su un valore `ShardId` esistente, ad esempio `shard-000000000004`, e seleziona Test.

    Il payload di risposta di esito positivo è simile all'output seguente:

    ```
    {
      "ShardIterator": "AAAAAAAAAAFYVN3VlFy..."
    }
    ```

Annota il valore di `ShardIterator`. Ti servirà per ottenere record da un flusso.

Per configurare e testare il metodo **`GET /streams/{stream-name}/records`** per richiamare l'operazione **`GetRecords`** in Kinesis

1. Scegli la risorsa /records, quindi scegli il metodo Create.
2. Per Tipo di metodo seleziona GET.
3. Per Tipo di integrazione seleziona Servizio AWS .
4. Per Regione AWS, seleziona il Regione AWS luogo in cui hai creato lo stream Kinesis.
5. Per Servizio AWS seleziona Kinesis.
6. Lascia vuoto Sottodominio AWS .
7. Per HTTP method (Metodo HTTP) scegli POST.
8. Per Tipo di operazione scegli Usa nome operazione.
9. Per Nome azione immetti **`GetRecords`**.
10. Per Ruolo di esecuzione immetti l'ARN del ruolo di esecuzione.
11. Lascia il valore predefinito Transito per Gestione contenuti.
12. Scegli le intestazioni delle richieste HTTP.

    L'operazione `GetRecords` richiede l'immissione di un valore `ShardIterator`. Per passare un `ShardIterator` valore fornito dal client, aggiungiamo un parametro di `Shard-Iterator` intestazione alla richiesta del metodo.
13. Seleziona Add header (Aggiungi intestazione).
14. Per Nome, immetti **`Shard-Iterator`**.
15. Mantieni Obbligatorio e Caching disattivati.
16. Scegli Crea metodo.
17. Nella sezione Richiesta di integrazione aggiungi il seguente modello di mappatura del corpo per mappare il valore del parametro di intestazione `Shard-Iterator` al valore della proprietà `ShardIterator` del payload JSON per l'azione `GetRecords` in Kinesis.

    ```
    {
        "ShardIterator": "$input.params('Shard-Iterator')"
    }
    ```

18. Usando l'opzione Test nella console Gateway API, immetti il nome di un flusso esistente come valore della variabile Percorso `stream-name`, imposta Intestazione `Shard-Iterator` sul valore `ShardIterator` ottenuto dall'esecuzione del test del metodo `GET /streams/{stream-name}/sharditerator` (precedente) e scegli Test.

    Il payload di risposta di esito positivo è simile all'output seguente:

```
{
  "MillisBehindLatest": 0,
  "NextShardIterator": "AAAAAAAAAAF...",
  "Records": [ ... ]
}
```

Definizioni OpenAPI di un'API di esempio come un proxy Kinesis

Di seguito sono riportate le definizioni OpenAPI per l'API di esempio come proxy Kinesis utilizzate in questo tutorial.

OpenAPI 3.0

```
{
  "openapi": "3.0.0",
  "info": {
    "title": "KinesisProxy",
    "version": "2016-03-31T18:25:32Z"
  },
  "paths": {
    "/streams/{stream-name}/sharditerator": {
      "get": {
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
          },
          {
            "name": "shard-id",
            "in": "query",
```

```
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/GetShardIterator",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"ShardId\": \"$input.params('shard-
id')\",\n    \"ShardIteratorType\": \"TRIM_HORIZON\",\n    \"StreamName\":
 \"$input.params('stream-name')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    },
    "/streams/{stream-name}/records": {
      "get": {
        "parameters": [
          {
```

```
            "name": "stream-name",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
          },
          {
            "name": "Shard-Iterator",
            "in": "header",
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/GetRecords",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"ShardIterator\": \"$input.params('Shard-
Iterator')\"\n}"
          },
```

```
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      },
      "put": {
        "parameters": [
          {
            "name": "Content-Type",
            "in": "header",
            "schema": {
              "type": "string"
            }
          },
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
          }
        ],
        "requestBody": {
          "content": {
            "application/json": {
              "schema": {
                "$ref": "#/components/schemas/PutRecordsMethodRequestPayload"
              }
            },
            "application/x-amz-json-1.1": {
              "schema": {
                "$ref": "#/components/schemas/PutRecordsMethodRequestPayload"
              }
            }
          },
          "required": true
        },
        "responses": {
          "200": {
            "description": "200 response",
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Empty"
```

```
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/PutRecords",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"$input.params('stream-
name')\",\n    \"Records\": [\n          {\n            \"Data\":
 \"$util.base64Encode($elem.data)\",\n            \"PartitionKey\":
 \"$elem.partition-key\"\n          }#if($foreach.hasNext),#end\n    ]\n}",
            "application/x-amz-json-1.1": "{\n  \"StreamName\":
 \"$input.params('stream-name')\",\n  \"records\" : [\n    {\n        \"Data
\" : \"$elem.data\",\n        \"PartitionKey\" : \"$elem.partition-key\"\n
  }#if($foreach.hasNext),#end\n  ]\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    },
    "/streams/{stream-name}": {
      "get": {
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
          }
```

```
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/DescribeStream",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"$input.params('stream-
name')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      },
      "post": {
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
```

```
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/CreateStream",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"ShardCount\": 5,\n    \"StreamName\":
 \"$input.params('stream-name')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      },
      "delete": {
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
```

```
            "headers": {
              "Content-Type": {
                "schema": {
                  "type": "string"
                }
              }
            },
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Empty"
                }
              }
            }
          },
          "400": {
            "description": "400 response",
            "headers": {
              "Content-Type": {
                "schema": {
                  "type": "string"
                }
              }
            },
            "content": {}
          },
          "500": {
            "description": "500 response",
            "headers": {
              "Content-Type": {
                "schema": {
                  "type": "string"
                }
              }
            },
            "content": {}
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/DeleteStream",
          "responses": {
            "4\\d{2}": {
```

```
              "statusCode": "400",
              "responseParameters": {
                "method.response.header.Content-Type":
 "integration.response.header.Content-Type"
              }
            },
            "default": {
              "statusCode": "200",
              "responseParameters": {
                "method.response.header.Content-Type":
 "integration.response.header.Content-Type"
              }
            },
            "5\\d{2}": {
              "statusCode": "500",
              "responseParameters": {
                "method.response.header.Content-Type":
 "integration.response.header.Content-Type"
              }
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"$input.params('stream-
name')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    },
    "/streams/{stream-name}/record": {
      "put": {
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
```

```
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/PutRecord",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"$input.params('stream-
name')\",\n    \"Data\": \"$util.base64Encode($input.json('$.Data'))\",\n
 \"PartitionKey\": \"$input.path('$.PartitionKey')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    },
    "/streams": {
      "get": {
        "responses": {
          "200": {
            "description": "200 response",
            "content": {
              "application/json": {
```

```
                "schema": {
                  "$ref": "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/ListStreams",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    }
  },
  "components": {
    "schemas": {
      "Empty": {
        "type": "object"
      },
      "PutRecordsMethodRequestPayload": {
        "type": "object",
        "properties": {
          "records": {
            "type": "array",
            "items": {
              "type": "object",
              "properties": {
                "data": {
                  "type": "string"
```

```
                },
                "partition-key": {
                  "type": "string"
                }
              }
            }
          }
        }
      }
    }
  }
}
```

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2016-03-31T18:25:32Z",
    "title": "KinesisProxy"
  },
  "basePath": "/test",
  "schemes": [
    "https"
  ],
  "paths": {
    "/streams": {
      "get": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
        },
        "x-amazon-apigateway-integration": {
```

```
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/ListStreams",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    },
    "/streams/{stream-name}": {
      "get": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
        },
```

```
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/DescribeStream",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"$input.params('stream-
name')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      },
      "post": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/CreateStream",
```

```
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"ShardCount\": 5,\n    \"StreamName\":
 \"$input.params('stream-name')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      },
      "delete": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            },
            "headers": {
              "Content-Type": {
                "type": "string"
              }
            }
          },
```

```
          "400": {
            "description": "400 response",
            "headers": {
              "Content-Type": {
                "type": "string"
              }
            }
          },
          "500": {
            "description": "500 response",
            "headers": {
              "Content-Type": {
                "type": "string"
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/DeleteStream",
          "responses": {
            "4\\d{2}": {
              "statusCode": "400",
              "responseParameters": {
                "method.response.header.Content-Type":
"integration.response.header.Content-Type"
              }
            },
            "default": {
              "statusCode": "200",
              "responseParameters": {
                "method.response.header.Content-Type":
"integration.response.header.Content-Type"
              }
            },
            "5\\d{2}": {
              "statusCode": "500",
              "responseParameters": {
                "method.response.header.Content-Type":
"integration.response.header.Content-Type"
              }
            }
          },
```

```
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"$input.params('stream-
name')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    },
    "/streams/{stream-name}/record": {
      "put": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/PutRecord",
          "responses": {
            "default": {
              "statusCode": "200"
```

```
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"$input.params('stream-
name')\",\n    \"Data\": \"$util.base64Encode($input.json('$.Data'))\",\n
 \"PartitionKey\": \"$input.path('$.PartitionKey')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    },
    "/streams/{stream-name}/records": {
      "get": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "type": "string"
          },
          {
            "name": "Shard-Iterator",
            "in": "header",
            "required": false,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
```

```
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/GetRecords",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"ShardIterator\": \"$input.params('Shard-
Iterator')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      },
      "put": {
        "consumes": [
          "application/json",
          "application/x-amz-json-1.1"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "Content-Type",
            "in": "header",
            "required": false,
            "type": "string"
          },
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "type": "string"
          },
```

```
          {
            "in": "body",
            "name": "PutRecordsMethodRequestPayload",
            "required": true,
            "schema": {
              "$ref": "#/definitions/PutRecordsMethodRequestPayload"
            }
          },
          {
            "in": "body",
            "name": "PutRecordsMethodRequestPayload",
            "required": true,
            "schema": {
              "$ref": "#/definitions/PutRecordsMethodRequestPayload"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/PutRecords",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"$input.params('stream-
name')\",\n    \"Records\": [\n          {\n            \"Data\":
 \"$util.base64Encode($elem.data)\",\n            \"PartitionKey\":
 \"$elem.partition-key\"\n          }#if($foreach.hasNext),#end\n    ]\n}",
```

```
            "application/x-amz-json-1.1": "{\n  \"StreamName\":
 \"$input.params('stream-name')\",\n  \"records\" : [\n    {\n        \"Data
\" : \"$elem.data\",\n        \"PartitionKey\" : \"$elem.partition-key\"\n
  }#if($foreach.hasNext),#end\n  ]\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    },
    "/streams/{stream-name}/sharditerator": {
      "get": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "type": "string"
          },
          {
            "name": "shard-id",
            "in": "query",
            "required": false,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/GetShardIterator",
```

```
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"ShardId\": \"$input.params('shard-
id')\",\n    \"ShardIteratorType\": \"TRIM_HORIZON\",\n    \"StreamName\":
 \"$input.params('stream-name')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    }
  },
  "definitions": {
    "Empty": {
      "type": "object"
    },
    "PutRecordsMethodRequestPayload": {
      "type": "object",
      "properties": {
        "records": {
          "type": "array",
          "items": {
            "type": "object",
            "properties": {
              "data": {
                "type": "string"
              },
              "partition-key": {
                "type": "string"
              }
            }
          }
        }
      }
    }
  }
```

```
}
```

Tutorial: crea un'API REST utilizzando AWS SDK o AWS CLI

Il seguente tutorial mostra come creare un' PetStore API che supporti i `GET /pets/{petId}` metodi `GET /pets` and. I metodi sono integrati con un endpoint HTTP. Puoi seguire questo tutorial usando l' AWS SDK for JavaScript, l'SDK for Python (Boto3) o il. AWS CLI Utilizzi le seguenti funzioni o comandi per configurare la tua API:

JavaScript v3

- CreateRestApiCommand
- CreateResourceCommand
- PutMethodCommand
- PutMethodResponseCommand
- PutIntegrationCommand
- PutIntegrationResponseCommand
- CreateDeploymentCommand

Python

- create_rest_api
- crea_risorsa
- metodo_put
- put_method_response
- put_integrazione
- put_integration_response
- creare_distribuzione

AWS CLI

- create-rest-api
- crea-risorsa
- metodo put

- put-method-response
- integrazione put
- put-integration-response
- creare-distribuire

Per ulteriori informazioni sull' AWS SDK per la versione JavaScript 3, vedi A cosa serve l'SDK? AWS JavaScript . Per ulteriori informazioni sull'SDK for Python (Boto3), consulta. AWS SDK for Python (Boto3) Per ulteriori informazioni su AWS CLI, consulta What is the? AWS CLI .

Configura un'API ottimizzata per l'edge PetStore

In questo tutorial, i comandi di esempio utilizzano valori segnaposto per ID di valore come ID API e ID risorsa. Man mano che completi il tutorial, sostituisci questi valori con i tuoi.

Per configurare un'API ottimizzata per i dispositivi periferici utilizzando gli PetStore SDK AWS

1. Usa l'esempio seguente per creare un'entità: `RestApi`

   JavaScript v3

```
import {APIGatewayClient, CreateRestApiCommand} from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new CreateRestApiCommand({
    name: "Simple PetStore (JavaScript v3 SDK)",
    description: "Demo API created using the AWS SDK for JavaScript v3",
    version: "0.00.001",
    binaryMediaTypes: [
    '*']
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.error(Couldn't create API:\n", err)
}
})();
```

Una chiamata riuscita restituisce l'ID API e l'ID della risorsa principale dell'API in un output come il seguente:

```
{
  id: 'abc1234',
  name: 'PetStore (JavaScript v3 SDK)',
  description: 'Demo API created using the AWS SDK for node.js',
  createdDate: 2017-09-05T19:32:35.000Z,
  version: '0.00.001',
  rootResourceId: 'efg567'
  binaryMediaTypes: [ '*' ]
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.create_rest_api(
        name='Simple PetStore (Python SDK)',
        description='Demo API created using the AWS SDK for Python',
        version='0.00.001',
        binaryMediaTypes=[
            '*'
        ]
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Couldn't create REST API %s.", error)
    raise
attribute=["id","name","description","createdDate","version","binaryMediaTypes","apiKeyS
filtered_result ={key:result[key] for key in attribute}
print(filtered_result)
```

Una chiamata riuscita restituisce l'ID API e l'ID della risorsa principale dell'API in un output come il seguente:

```
{'id': 'abc1234', 'name': 'Simple PetStore (Python SDK)', 'description':
 'Demo API created using the AWS SDK for Python', 'createdDate':
 datetime.datetime(2024, 4, 3, 14, 31, 39, tzinfo=tzlocal()), 'version':
 '0.00.001', 'binaryMediaTypes': ['*'], 'apiKeySource': 'HEADER',
 'endpointConfiguration': {'types': ['EDGE']}, 'disableExecuteApiEndpoint':
 False, 'rootResourceId': 'efg567'}
```

AWS CLI

```
aws apigateway create-rest-api --name 'Simple PetStore (AWS CLI)' --region us-
west-2
```

L'output di questo comando è il seguente:

```
{
    "id": "abcd1234",
    "name": "Simple PetStore (AWS CLI)",
    "createdDate": "2022-12-15T08:07:04-08:00",
    "apiKeySource": "HEADER",
    "endpointConfiguration": {
        "types": [
            "EDGE"
        ]
    },
    "disableExecuteApiEndpoint": false,
    "rootResourceId": "efg567"
}
```

L'API che hai creato ha un ID API di `abcd1234` e un ID di risorsa root die`fg567`. Utilizzi questi valori nella configurazione della tua API.

2. Successivamente, aggiungi una risorsa figlia sotto la radice, la specifichi `RootResourceId` come valore della `parentId` proprietà. Utilizzate l'esempio seguente per creare una `/pets` risorsa per la vostra API:

   JavaScript v3

```
import {APIGatewayClient,  CreateResourceCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
```

```
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new CreateResourceCommand({
    restApiId: 'abcd1234',
    parentId: 'efg567',
    pathPart: 'pets'
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("The '/pets' resource setup failed:\n", err)
}
})();
```

Una chiamata riuscita restituisce informazioni sulla risorsa in un output come il seguente:

```
{
    "path": "/pets",
    "pathPart": "pets",
    "id": "aaa111",
    "parentId": "efg567'"
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.create_resource(
        restApiId='abcd1234',
        parentId='efg567',
        pathPart='pets'
    )
except botocore.exceptions.ClientError as error:
    logger.exception("The '/pets' resource setup failed: %s.", error)
    raise
attribute=["id","parentId", "pathPart", "path",]
```

```
filtered_result ={key:result[key] for key in attribute}
print(filtered_result)
```

Una chiamata riuscita restituisce informazioni sulla risorsa in un output come il seguente:

```
{'id': 'aaa111', 'parentId': 'efg567', 'pathPart': 'pets', 'path': '/pets'}
```

AWS CLI

```
aws apigateway create-resource --rest-api-id abcd1234 \
  --region us-west-2 \
  --parent-id efg567 \
  --path-part pets
```

L'output di questo comando è il seguente:

```
{
    "id": "aaa111",
    "parentId": "efg567",
    "pathPart": "pets",
    "path": "/pets"
}
```

La `/pets` risorsa che hai creato ha un ID di risorsa di`aaa111`. Utilizzi questo valore nella configurazione della tua API.

3. Successivamente, aggiungi una risorsa secondaria sotto la `/pets` risorsa. Questa risorsa `/{petId}` ha un parametro path per `{petId}` .Per rendere una parte del percorso un parametro di percorso, racchiudetelo tra parentesi graffe,. `{ }` Usa l'esempio seguente per creare una `/pets/{petId}` risorsa per la tua API:

   JavaScript v3

```
import {APIGatewayClient,  CreateResourceCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new CreateResourceCommand({
    restApiId: 'abcd1234',
    parentId: 'aaa111',
```

```
    pathPart: '{petId}'
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("The '/pets/{petId}' resource setup failed:\n", err)
}
})();
```

Una chiamata riuscita restituisce informazioni sulla risorsa in un output come il seguente:

```
{
    "path": "/pets/{petId}",
    "pathPart": "{petId}",
    "id": "bbb222",
    "parentId": "aaa111'"
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.create_resource(
        restApiId='abcd1234',
        parentId='aaa111',
        pathPart='{petId}'
    )
except botocore.exceptions.ClientError as error:
    logger.exception("The '/pets/{petId}' resource setup failed: %s.", error)
    raise
attribute=["id","parentId", "pathPart", "path",]
filtered_result ={key:result[key] for key in attribute}
print(filtered_result)
```

Una chiamata riuscita restituisce informazioni sulla risorsa in un output come il seguente:

```
{'id': 'bbb222', 'parentId': 'aaa111', 'pathPart': '{petId}', 'path': '/pets/
{petId}'}
```

AWS CLI

```
aws apigateway create-resource --rest-api-id abcd1234 \
  --region us-west-2 \
  --parent-id aaa111 \
  --path-part '{petId}'
```

In caso di esito positivo, il comando restituisce la risposta seguente:

```
{
    "id": "bbb222",
    "parentId": "aaa111",
    "path": "/pets/{petId}",
    "pathPart": "{petId}"
}
```

La `/pets/{petId}` risorsa che hai creato ha un ID di risorsa dibbb222. Utilizzi questo valore nella configurazione della tua API.

4. Nei due passaggi seguenti, aggiungi metodi HTTP alle tue risorse. In questo tutorial, imposterai i metodi per avere accesso aperto impostando `authorization-type` l'impostazione su`NONE`. Per consentire solo agli utenti autenticati di chiamare il metodo, puoi utilizzare i ruoli e le policy IAM, un'autorizzazione Lambda (nota in precedenza come autorizzazioni ad hoc) o un pool di utenti di Amazon Cognito. Per ulteriori informazioni, consulta the section called “Controllo accessi”.

   L'esempio seguente aggiunge il metodo `GET` HTTP alla `/pets` risorsa:

   JavaScript v3

```
import {APIGatewayClient,  PutMethodCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new PutMethodCommand({
    restApiId: 'abcd1234',
```

```
    resourceId: 'aaa111',
    httpMethod: 'GET',
    authorizationType: 'NONE'
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("The 'GET /pets' method setup failed:\n", err)
}
})();
```

Una chiamata riuscita restituisce il seguente risultato:

```
{
    "apiKeyRequired": false,
    "httpMethod": "GET",
    "authorizationType": "NONE"
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.put_method(
        restApiId='abcd1234',
        resourceId='aaa111',
        httpMethod='GET',
        authorizationType='NONE'
    )
except botocore.exceptions.ClientError as error:
    logger.exception("The 'GET /pets' method setup failed: %s", error)
    raise
attribute=["httpMethod","authorizationType","apiKeyRequired"]
filtered_result ={key:result[key] for key in attribute}
print(filtered_result)
```

Una chiamata riuscita restituisce il seguente risultato:

```
{'httpMethod': 'GET', 'authorizationType': 'NONE', 'apiKeyRequired': False}
```

AWS CLI

```
aws apigateway put-method --rest-api-id abcd1234 \
  --resource-id aaa111 \
  --http-method GET \
  --authorization-type "NONE" \
  --region us-west-2
```

Se il comando riesce, l'output è il seguente:

```
{
    "httpMethod": "GET",
    "authorizationType": "NONE",
    "apiKeyRequired": false
}
```

5. L'esempio seguente aggiunge il metodo `GET` HTTP alla `/pets/{petId}` risorsa e imposta la `requestParameters` proprietà per passare il `petId` valore fornito dal client al backend:

JavaScript v3

```
import {APIGatewayClient,  PutMethodCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new PutMethodCommand({
    restApiId: 'abcd1234',
    resourceId: 'bbb222',
    httpMethod: 'GET',
    authorizationType: 'NONE'
    requestParameters: {
        "method.request.path.petId" : true
    }
});
try {
    const results = await apig.send(command)
    console.log(results)
```

```
} catch (err) {
    console.log("The 'GET /pets/{petId}' method setup failed:\n", err)
}
})();
```

Una chiamata riuscita restituisce il seguente risultato:

```
{
    "apiKeyRequired": false,
    "httpMethod": "GET",
    "authorizationType": "NONE",
    "requestParameters": {
       "method.request.path.petId": true
    }
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.put_method(
        restApiId='abcd1234',
        resourceId='bbb222',
        httpMethod='GET',
        authorizationType='NONE',
        requestParameters={
            "method.request.path.petId": True
        }
    )
except botocore.exceptions.ClientError as error:
    logger.exception("The 'GET /pets/{petId}' method setup failed: %s", error)
    raise
attribute=["httpMethod","authorizationType","apiKeyRequired",
 "requestParameters" ]
filtered_result ={key:result[key] for key in attribute}
print(filtered_result)
```

Una chiamata riuscita restituisce il seguente risultato:

```
{'httpMethod': 'GET', 'authorizationType': 'NONE', 'apiKeyRequired': False,
 'requestParameters': {'method.request.path.petId': True}}
```

AWS CLI

```
aws apigateway put-method --rest-api-id abcd1234 \
  --resource-id bbb222 --http-method GET \
  --authorization-type "NONE" \
  --region us-west-2 \
  --request-parameters method.request.path.petId=true
```

Se il comando riesce, l'output è il seguente:

```
{
    "httpMethod": "GET",
    "authorizationType": "NONE",
    "apiKeyRequired": false,
    "requestParameters": {
        "method.request.path.petId": true
    }
}
```

6. Utilizzate l'esempio seguente per aggiungere la risposta del metodo 200 OK per il `GET /pets` metodo:

JavaScript v3

```
import {APIGatewayClient,  PutMethodResponseCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new PutMethodResponseCommand({
    restApiId: 'abcd1234',
    resourceId: 'aaa111',
    httpMethod: 'GET',
    statusCode: '200'
});
try {
    const results = await apig.send(command)
```

```
    console.log(results)
} catch (err) {
    console.log("Set up the 200 OK response for the 'GET /pets' method failed:
\n", err)
}
})();
```

Una chiamata riuscita restituisce il seguente risultato:

```
{
    "statusCode": "200"
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.put_method_response(
        restApiId='abcd1234',
        resourceId='aaa111',
        httpMethod='GET',
        statusCode='200'
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Set up the 200 OK response for the 'GET /pets' method
 failed %s.", error)
    raise
attribute=["statusCode"]
filtered_result ={key:result[key] for key in attribute}
logger.info(filtered_result)
```

Una chiamata riuscita restituisce il seguente risultato:

```
{'statusCode': '200'}
```

AWS CLI

```
aws apigateway put-method-response --rest-api-id abcd1234 \
  --resource-id aaa111 --http-method GET \
  --status-code 200  --region us-west-2
```

L'output di questo comando è il seguente:

```
{
    "statusCode": "200"
}
```

7. Utilizzate l'esempio seguente per aggiungere la risposta del metodo 200 OK per il `GET /pets/{petId}` metodo:

JavaScript v3

```
import {APIGatewayClient,  PutMethodResponseCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new PutMethodResponseCommand({
    restApiId: 'abcd1234',
    resourceId: 'bbb222',
    httpMethod: 'GET',
    statusCode: '200'
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("Set up the 200 OK response for the 'GET /pets/{petId}' method
 failed:\n", err)
}
})();
```

Una chiamata riuscita restituisce il seguente risultato:

```
{
    "statusCode": "200"
```

```
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.put_method_response(
        restApiId='abcd1234',
        resourceId='bbb222',
        httpMethod='GET',
        statusCode='200'
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Set up the 200 OK response for the 'GET /pets/{petId}'
 method failed %s.", error)
    raise
attribute=["statusCode"]
filtered_result ={key:result[key] for key in attribute}
logger.info(filtered_result)
```

Una chiamata riuscita restituisce il seguente risultato:

```
{'statusCode': '200'}
```

AWS CLI

```
aws apigateway put-method-response --rest-api-id abcd1234 \
  --resource-id bbb222 --http-method GET \
  --status-code 200  --region us-west-2
```

L'output di questo comando è il seguente:

```
{
    "statusCode": "200"
}
```

8. L'esempio seguente configura un'integrazione per il `GET /pets` metodo con un endpoint HTTP. L'endpoint HTTP è. `http://petstore-demo-endpoint.execute-api.com/petstore/pets`

   JavaScript v3

```
import {APIGatewayClient,  PutIntegrationCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new PutIntegrationCommand({
    restApiId: 'abcd1234',
    resourceId: 'aaa111',
    httpMethod: 'GET',
    type: 'HTTP',
    integrationHttpMethod: 'GET',
    uri: 'http://petstore-demo-endpoint.execute-api.com/petstore/pets'
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("Set up the integration of the 'GET /pets' method of the API
 failed:\n", err)
}
})();
```

   Una chiamata riuscita restituisce il seguente risultato:

```
{
    "httpMethod": "GET",
    "passthroughBehavior": "WHEN_NO_MATCH",
    "cacheKeyParameters": [],
    "type": "HTTP",
    "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
    "cacheNamespace": "ccc333"
}
```

   Python

```
import botocore
import boto3
```

```
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.put_integration(
        restApiId='abcd1234',
        resourceId='aaa111',
        httpMethod='GET',
        type='HTTP',
        integrationHttpMethod='GET',
        uri='http://petstore-demo-endpoint.execute-api.com/petstore/pets'
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Set up the integration of the 'GET /' method of the API
 failed %s.", error)
    raise
attribute=["httpMethod","passthroughBehavior","cacheKeyParameters", "type",
 "uri", "cacheNamespace"]
filtered_result ={key:result[key] for key in attribute}
print(filtered_result)
```

Una chiamata riuscita restituisce il seguente risultato:

```
{'httpMethod': 'GET', 'passthroughBehavior': 'WHEN_NO_MATCH',
 'cacheKeyParameters': [], 'type': 'HTTP', 'uri': 'http://petstore-demo-
endpoint.execute-api.com/petstore/pets', 'cacheNamespace': 'ccc333'}
```

AWS CLI

```
aws apigateway put-integration --rest-api-id abcd1234 \
  --resource-id aaa111 --http-method GET --type HTTP \
  --integration-http-method GET \
  --uri 'http://petstore-demo-endpoint.execute-api.com/petstore/pets' \
  --region us-west-2
```

L'output di questo comando è il seguente:

```
{
    "type": "HTTP",
    "httpMethod": "GET",
```

```
    "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
    "connectionType": "INTERNET",
    "passthroughBehavior": "WHEN_NO_MATCH",
    "timeoutInMillis": 29000,
    "cacheNamespace": "6sxz2j",
    "cacheKeyParameters": []
}
```

9. L'esempio seguente configura un'integrazione per il `GET /pets/{petId}` metodo con un endpoint HTTP. L'endpoint HTTP è. `http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}` In questo passaggio, si mappa il parametro path `petId` al parametro del percorso dell'endpoint di integrazione di. `id`

   JavaScript v3

```
import {APIGatewayClient,  PutIntegrationCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new PutIntegrationCommand({
    restApiId: 'abcd1234',
    resourceId: 'bbb222',
    httpMethod: 'GET',
    type: 'HTTP',
    integrationHttpMethod: 'GET',
    uri: 'http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}'
    requestParameters: {
        "integration.request.path.id": "method.request.path.petId"
     }
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("Set up the integration of the 'GET /pets/{petId}' method of the
 API failed:\n", err)
}
})();
```

   Una chiamata riuscita restituisce il seguente risultato:

```
{
```

```
    "httpMethod": "GET",
    "passthroughBehavior": "WHEN_NO_MATCH",
    "cacheKeyParameters": [],
    "type": "HTTP",
    "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}",
    "cacheNamespace": "ddd444",
    "requestParameters": {
       "integration.request.path.id": "method.request.path.petId"
    }
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.put_integration(
        restApiId='ieps9b05sf',
        resourceId='t8zeb4',
        httpMethod='GET',
        type='HTTP',
        integrationHttpMethod='GET',
        uri='http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}',
        requestParameters={
            "integration.request.path.id": "method.request.path.petId"
        }
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Set up the integration of the 'GET /pets/{petId}' method
 of the API failed %s.", error)
    raise
attribute=["httpMethod","passthroughBehavior","cacheKeyParameters", "type",
 "uri", "cacheNamespace", "requestParameters"]
filtered_result ={key:result[key] for key in attribute}
print(filtered_result)
```

Una chiamata riuscita restituisce il seguente risultato:

```
{'httpMethod': 'GET', 'passthroughBehavior': 'WHEN_NO_MATCH',
 'cacheKeyParameters': [], 'type': 'HTTP', 'uri': 'http://petstore-
demo-endpoint.execute-api.com/petstore/pets/{id}', 'cacheNamespace':
 'ddd444', 'requestParameters': {'integration.request.path.id':
 'method.request.path.petId'}}}
```

AWS CLI

```
aws apigateway put-integration --rest-api-id abcd1234 \
  --resource-id bbb222 --http-method GET --type HTTP \
  --integration-http-method GET \
  --uri 'http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}' \
  --request-parameters
 '{"integration.request.path.id":"method.request.path.petId"}' \
  --region us-west-2
```

L'output di questo comando è il seguente:

```
{
    "type": "HTTP",
    "httpMethod": "GET",
    "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}",
    "connectionType": "INTERNET",
    "requestParameters": {
        "integration.request.path.id": "method.request.path.petId"
    },
    "passthroughBehavior": "WHEN_NO_MATCH",
    "timeoutInMillis": 29000,
    "cacheNamespace": "rjkmth",
    "cacheKeyParameters": []
}
```

10. L'esempio seguente aggiunge la risposta di integrazione per l'`GET /pets`integrazione:

    JavaScript v3

```
import {APIGatewayClient,  PutIntegrationResponseCommand } from "@aws-sdk/
client-api-gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new PutIntegrationResponseCommand({
```

```
    restApiId: 'abcd1234',
    resourceId: 'aaa111',
    httpMethod: 'GET',
    statusCode: '200',
    selectionPattern: ''
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("The 'GET /pets' method integration response setup failed:\n",
 err)
}
})();
```

Una chiamata riuscita restituisce il seguente risultato:

```
{
    "selectionPattern": "",
    "statusCode": "200"
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.put_integration_response(
        restApiId='abcd1234',
        resourceId='aaa111',
        httpMethod='GET',
        statusCode='200',
        selectionPattern='',
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Set up the integration response of the 'GET /pets' method
 of the API failed: %s", error)
```

```
    raise
attribute=["selectionPattern","statusCode"]
filtered_result ={key:result[key] for key in attribute}
print(filtered_result)
```

Una chiamata riuscita restituisce il seguente risultato:

```
{'selectionPattern': "", 'statusCode': '200'}
```

AWS CLI

```
aws apigateway put-integration-response --rest-api-id abcd1234 \
  --resource-id aaa111 --http-method GET \
  --status-code 200 --selection-pattern ""  \
  --region us-west-2
```

L'output di questo comando è il seguente:

```
{
    "statusCode": "200",
    "selectionPattern": ""
}
```

11. L'esempio seguente aggiunge la risposta di integrazione per l'`GET /pets/{petId}`integrazione:

    JavaScript v3

```
import {APIGatewayClient,  PutIntegrationResponseCommand } from "@aws-sdk/
client-api-gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new PutIntegrationResponseCommand({
    restApiId: 'abcd1234',
    resourceId: 'bbb222',
    httpMethod: 'GET',
    statusCode: '200',
    selectionPattern: ''
});
try {
    const results = await apig.send(command)
```

```
    console.log(results)
} catch (err) {
    console.log("The 'GET /pets/{petId}' method integration response setup
 failed:\n", err)
}
})();
```

Una chiamata riuscita restituisce il seguente risultato:

```
{
    "selectionPattern": "",
    "statusCode": "200"
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.put_integration_response(
        restApiId='abcd1234',
        resourceId='bbb222',
        httpMethod='GET',
        statusCode='200',
        selectionPattern='',
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Set up the integration response of the 'GET /pets/{petId}'
 method of the API failed: %s", error)
    raise
attribute=["selectionPattern","statusCode"]
filtered_result ={key:result[key] for key in attribute}
print(filtered_result)
```

Una chiamata riuscita restituisce il seguente risultato:

```
{'selectionPattern': "", 'statusCode': '200'}
```

AWS CLI

```
aws apigateway put-integration-response --rest-api-id abcd1234 \
  --resource-id bbb222 --http-method GET
  --status-code 200 --selection-pattern ""
  --region us-west-2
```

L'output di questo comando è il seguente:

```
{
    "statusCode": "200",
    "selectionPattern": ""
}
```

Dopo aver creato la risposta di integrazione, l'API può interrogare gli animali domestici disponibili sul PetStore sito Web e visualizzare un singolo animale domestico con un identificatore specificato. Prima che l'API sia richiamabile dai clienti, devi implementarla. Ti consigliamo di testarla prima di distribuire l'API.

12. L'esempio seguente verifica il `GET /pets` metodo:

JavaScript v3

```
import {APIGatewayClient,  TestInvokeMethodCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new TestInvokeMethodCommand({
    restApiId: 'abcd1234',
    resourceId: 'aaa111',
    httpMethod: 'GET',
    pathWithQueryString: '/',
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("The test on 'GET /pets' method failed:\n", err)
```

```
}
})();
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.test_invoke_method(
        restApiId='abcd1234',
        resourceId='aaa111',
        httpMethod='GET',
        pathWithQueryString='/',
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Test invoke method on 'GET /pets' failed: %s", error)
    raise
print(result)
```

AWS CLI

```
aws apigateway test-invoke-method --rest-api-id abcd1234 /
  --resource-id aaa111 /
  --http-method GET /
  --path-with-query-string '/'
```

13. L'esempio seguente verifica il `GET /pets/{petId}` metodo con un punteggio `petId` di 3:

JavaScript v3

```
import {APIGatewayClient,  TestInvokeMethodCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new TestInvokeMethodCommand({
    restApiId: 'abcd1234',
    resourceId: 'bbb222',
```

```
    httpMethod: 'GET',
    pathWithQueryString: '/pets/3',
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("The test on 'GET /pets/{petId}' method failed:\n", err)
}
})();
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.test_invoke_method(
        restApiId='abcd1234',
        resourceId='bbb222',
        httpMethod='GET',
        pathWithQueryString='/pets/3',
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Test invoke method on 'GET /pets/{petId}' failed: %s",
 error)
    raise
print(result)
```

AWS CLI

```
aws apigateway test-invoke-method --rest-api-id abcd1234 /
  --resource-id bbb222 /
  --http-method GET /
  --path-with-query-string '/pets/3'
```

Dopo aver testato con successo l'API, puoi distribuirla in una fase successiva.

14. L'esempio seguente distribuisce l'API in una fase denominata. `test` Quando distribuisci l'API in una fase, i chiamanti dell'API possono richiamarla.

JavaScript v3

```
import {APIGatewayClient,  CreateDeploymentCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new CreateDeploymentCommand({
    restApiId: 'abcd1234',
    stageName: 'test',
    stageDescription: 'test deployment'
});
try {
    const results = await apig.send(command)
    console.log("Deploying API succeeded\n", results)
} catch (err) {
    console.log("Deploying API failed:\n", err)
}
})();
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.create_deployment(
        restApiId='ieps9b05sf',
        stageName='test',
        stageDescription='my test stage',
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Error deploying stage  %s.", error)
    raise
print('Deploying API succeeded')
print(result)
```

AWS CLI

```
aws apigateway create-deployment --rest-api-id abcd1234 \
  --region us-west-2 \
  --stage-name test \
  --stage-description 'Test stage' \
  --description 'First deployment'
```

L'output di questo comando è il seguente:

```
{
    "id": "ab1c1d",
    "description": "First deployment",
    "createdDate": "2022-12-15T08:44:13-08:00"
}
```

La tua API è ora richiamabile dai clienti. Puoi testare questa API inserendo l'`https://abcd1234.execute-api.us-west-2.amazonaws.com/test/pets`URL in un browser e sostituendolo `abcd1234` con l'identificatore dell'API.

Per altri esempi su come creare o aggiornare un'API utilizzando gli AWS SDK o il AWS CLI, consulta Actions for API Gateway using AWS SDK.

Automatizza la configurazione della tua API

Invece di creare la tua API step-by-step, puoi automatizzare la creazione e la pulizia delle AWS risorse utilizzando OpenAPI o Terraform per creare AWS CloudFormation la tua API.

Definizione OpenAPI 3.0

È possibile importare una definizione OpenAPI in API Gateway. Per ulteriori informazioni, consulta the section called “Apri API”.

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "Simple PetStore (OpenAPI)",
    "description" : "Demo API created using OpenAPI",
    "version" : "2024-05-24T20:39:34Z"
```

```
  },
  "servers" : [ {
    "url" : "{basePath}",
    "variables" : {
      "basePath" : {
        "default" : "Prod"
      }
    }
  } ],
  "paths" : {
    "/pets" : {
      "get" : {
        "responses" : {
          "200" : {
            "description" : "200 response",
            "content" : { }
          }
        },
        "x-amazon-apigateway-integration" : {
          "type" : "http",
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
          "responses" : {
            "default" : {
              "statusCode" : "200"
            }
          },
          "passthroughBehavior" : "when_no_match",
          "timeoutInMillis" : 29000
        }
      }
    },
    "/pets/{petId}" : {
      "get" : {
        "parameters" : [ {
          "name" : "petId",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "200" : {
```

```
            "description" : "200 response",
            "content" : { }
          }
        },
        "x-amazon-apigateway-integration" : {
          "type" : "http",
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}",
          "responses" : {
            "default" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.path.id" : "method.request.path.petId"
          },
          "passthroughBehavior" : "when_no_match",
          "timeoutInMillis" : 29000
        }
      }
    }
  },
  "components" : { }
}
```

AWS CloudFormation modello

Per distribuire il AWS CloudFormation modello, consulta Creazione di uno stack sulla AWS CloudFormation console.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Name: Simple PetStore (AWS CloudFormation)
  PetsResource:
    Type: 'AWS::ApiGateway::Resource'
    Properties:
      RestApiId: !Ref Api
      ParentId: !GetAtt Api.RootResourceId
      PathPart: 'pets'
  PetIdResource:
```

```
    Type: 'AWS::ApiGateway::Resource'
    Properties:
      RestApiId: !Ref Api
      ParentId: !Ref PetsResource
      PathPart: '{petId}'
  PetsMethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref PetsResource
      HttpMethod: GET
      AuthorizationType: NONE
      Integration:
        Type: HTTP
        IntegrationHttpMethod: GET
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
        IntegrationResponses:
          - StatusCode: '200'
      MethodResponses:
        - StatusCode: '200'
  PetIdMethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref PetIdResource
      HttpMethod: GET
      AuthorizationType: NONE
      RequestParameters:
        method.request.path.petId: true
      Integration:
        Type: HTTP
        IntegrationHttpMethod: GET
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}
        RequestParameters:
          integration.request.path.id: method.request.path.petId
        IntegrationResponses:
          - StatusCode: '200'
      MethodResponses:
        - StatusCode: '200'
  ApiDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn:
      - PetsMethodGet
    Properties:
```

```
      RestApiId: !Ref Api
      StageName: Prod
Outputs:
  ApiRootUrl:
    Description: Root Url of the API
    Value: !Sub 'https://${Api}.execute-api.${AWS::Region}.amazonaws.com/Prod'
```

Configurazione Terraform

Per ulteriori informazioni su Terraform, vedere Terraform.

```
provider "aws" {
  region = "us-east-1" # Update with your desired region
}
resource "aws_api_gateway_rest_api" "Api" {
  name        = "Simple PetStore (Terraform)"
  description = "Demo API created using Terraform"
}
resource "aws_api_gateway_resource" "petsResource"{
    rest_api_id = aws_api_gateway_rest_api.Api.id
    parent_id = aws_api_gateway_rest_api.Api.root_resource_id
    path_part = "pets"
}
resource "aws_api_gateway_resource" "petIdResource"{
    rest_api_id = aws_api_gateway_rest_api.Api.id
    parent_id = aws_api_gateway_resource.petsResource.id
    path_part = "{petId}"
}
resource "aws_api_gateway_method" "petsMethodGet" {
  rest_api_id   = aws_api_gateway_rest_api.Api.id
  resource_id   = aws_api_gateway_resource.petsResource.id
  http_method   = "GET"
  authorization = "NONE"
}


resource "aws_api_gateway_method_response" "petsMethodResponseGet" {
    rest_api_id = aws_api_gateway_rest_api.Api.id
    resource_id = aws_api_gateway_resource.petsResource.id
    http_method = aws_api_gateway_method.petsMethodGet.http_method
    status_code ="200"
}

resource "aws_api_gateway_integration" "petsIntegration" {
```

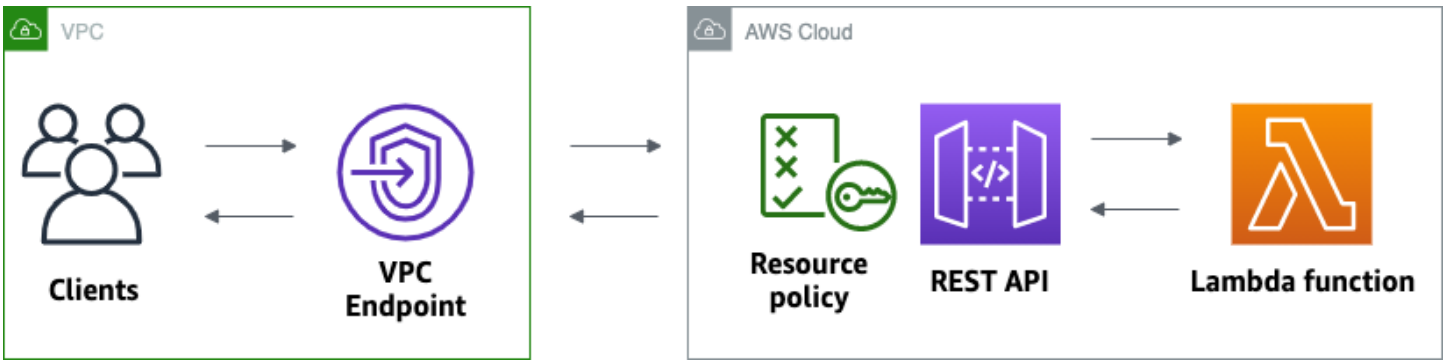
```
  rest_api_id = aws_api_gateway_rest_api.Api.id
  resource_id = aws_api_gateway_resource.petsResource.id
  http_method = aws_api_gateway_method.petsMethodGet.http_method
  type        = "HTTP"

  uri                     = "http://petstore-demo-endpoint.execute-api.com/petstore/
pets"
  integration_http_method = "GET"
  depends_on              = [aws_api_gateway_method.petsMethodGet]
}

resource "aws_api_gateway_integration_response" "petsIntegrationResponse" {
    rest_api_id = aws_api_gateway_rest_api.Api.id
    resource_id = aws_api_gateway_resource.petsResource.id
    http_method = aws_api_gateway_method.petsMethodGet.http_method
    status_code = aws_api_gateway_method_response.petsMethodResponseGet.status_code
}

resource "aws_api_gateway_method" "petIdMethodGet" {
    rest_api_id   = aws_api_gateway_rest_api.Api.id
    resource_id   = aws_api_gateway_resource.petIdResource.id
    http_method   = "GET"
    authorization = "NONE"
    request_parameters = {"method.request.path.petId" = true}
}

resource "aws_api_gateway_method_response" "petIdMethodResponseGet" {
    rest_api_id = aws_api_gateway_rest_api.Api.id
    resource_id = aws_api_gateway_resource.petIdResource.id
    http_method = aws_api_gateway_method.petIdMethodGet.http_method
    status_code ="200"
}


resource "aws_api_gateway_integration" "petIdIntegration" {
    rest_api_id = aws_api_gateway_rest_api.Api.id
    resource_id = aws_api_gateway_resource.petIdResource.id
    http_method = aws_api_gateway_method.petIdMethodGet.http_method
    type        = "HTTP"
    uri                     = "http://petstore-demo-endpoint.execute-api.com/petstore/
pets/{id}"
    integration_http_method = "GET"
    request_parameters = {"integration.request.path.id" = "method.request.path.petId"}
    depends_on              = [aws_api_gateway_method.petIdMethodGet]
```

```
}

resource "aws_api_gateway_integration_response" "petIdIntegrationResponse" {
    rest_api_id = aws_api_gateway_rest_api.Api.id
    resource_id = aws_api_gateway_resource.petIdResource.id
    http_method = aws_api_gateway_method.petIdMethodGet.http_method
    status_code = aws_api_gateway_method_response.petIdMethodResponseGet.status_code
}


resource "aws_api_gateway_deployment" "Deployment" {
  rest_api_id = aws_api_gateway_rest_api.Api.id
  depends_on  =
 [aws_api_gateway_integration.petsIntegration,aws_api_gateway_integration.petIdIntegration ]
}
resource "aws_api_gateway_stage" "Stage" {
  stage_name    = "Prod"
  rest_api_id   = aws_api_gateway_rest_api.Api.id
  deployment_id = aws_api_gateway_deployment.Deployment.id
}
```

Tutorial: crea un account privato REST API

In questo tutorial, crei un file privato RESTAPI. I clienti possono accedere API solo dall'interno del tuo AmazonVPC. APIÈ isolato dalla rete Internet pubblica, che è un requisito di sicurezza comune.

Il completamento di questa esercitazione richiede circa 30 minuti. Innanzitutto, utilizzi un AWS CloudFormation modello per creare un AmazonVPC, un VPC endpoint, una AWS Lambda funzione e avviare un'EC2istanza Amazon che utilizzerai per testare la tuaAPI. Successivamente, usi il AWS Management Console per creare una politica privata API e allegare una politica delle risorse che consenta l'accesso solo dal tuo VPC endpoint. Infine, metti alla prova il tuo. API



Per completare questo tutorial, sono necessari un AWS account e un AWS Identity and Access Management utente con accesso alla console. Per ulteriori informazioni, consulta Prerequisiti.

In questo tutorial utilizzerai AWS Management Console. Per un AWS CloudFormation modello che crea questa API e tutte le risorse correlate, vedi template.yaml.

Argomenti

- Fase 1: Creare dipendenze
- Passaggio 2: crea un account privato API
- Fase 3: Creare un metodo e un'integrazione
- Fase 4: Allegare una policy sulle risorse
- Fase 5: Implementa il tuo API
- Passo 6: Verifica che il tuo API non sia accessibile al pubblico
- Passaggio 7: Connettiti a un'istanza nel tuo VPC e richiama il API
- Fase 8: Pulizia
- Passaggi successivi: automatizza con AWS CloudFormation

Fase 1: Creare dipendenze

Scarica e decomprimi questo modello. AWS CloudFormation Utilizzi il modello per creare tutte le dipendenze per il tuo ambiente privatoAPI, tra cui un AmazonVPC, un VPC endpoint e una funzione Lambda che funge da backend del tuo. API Il privato lo crei in un secondo momento. API

Per creare una AWS CloudFormation pila

1. Apri la AWS CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformation.
2. Scegliere Create stack (Crea stack), quindi With new resources (standard) (Con nuove risorse (standard)).
3. In Specificare modello, scegliere Carica un file modello.
4. Selezionare il modello scaricato.
5. Seleziona Successivo.
6. Per Stack name (Nome stack), inserire **`private-api-tutorial`**, quindi scegliere Next (Avanti).

7. Per Configure stack options (Configura opzioni di stack), scegliere Next (Successivo).
8. Per quanto riguarda le funzionalità, riconosci che AWS CloudFormation puoi creare IAM risorse nel tuo account.
9. Scegli Invia.

AWS CloudFormation fornisce le tue dipendenzeAPI, operazione che può richiedere alcuni minuti. Quando lo stato del tuo AWS CloudFormation stack è CREATE_ COMPLETE, scegli Outputs. Prendi nota del tuo ID VPC endpoint. È necessario per le fasi successive di questo tutorial.

Passaggio 2: crea un account privato API

Crei un account privato API per consentire l'accesso solo VPC ai tuoi clienti.

Per creare un account privato API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli Crea API, quindi scegli Build. RESTAPI
3. Per APInome, inserisci**`private-api-tutorial`**.
4. Per il tipo di API endpoint, seleziona Privato.
5. Per VPCendpoint IDs, inserisci l'ID dell'VPCendpoint dagli output del tuo stack. AWS CloudFormation
6. Scegli Crea. API

Fase 3: Creare un metodo e un'integrazione

Crei un `GET` metodo e un'integrazione Lambda per gestire `GET` le richieste al tuo. API Quando un client richiama la tuaAPI, API Gateway invia la richiesta alla funzione Lambda creata nel passaggio 1, quindi restituisce una risposta al client.

Per creare un metodo e un'integrazione

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli il tuo API.
3. Scegli Crea metodo.
4. Per Tipo di metodo seleziona `GET`.
5. Per Tipo di integrazione seleziona Funzione Lambda.

6. Attiva l'opzione Integrazione proxy Lambda. Con l'integrazione del proxy Lambda, API Gateway invia un evento a Lambda con una struttura definita e trasforma la risposta dalla funzione Lambda in una risposta. HTTP
7. Per la funzione Lambda, scegli la funzione che hai creato con il AWS CloudFormation modello nel passaggio 1. Il nome della funzione inizia con **`private-api-tutorial`**.
8. Scegli Crea metodo.

Fase 4: Allegare una policy sulle risorse

Alleghi alla tua una politica delle risorse API che consente ai clienti di richiamarti API solo tramite il tuo endpoint. VPC Per limitare ulteriormente l'accesso al tuo API dispositivo, puoi anche configurare una VPCpolicy per gli VPC endpoint, ma non è necessario per questo tutorial.

Per allegare una policy sulle risorse

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli il tuo API.
3. Scegli Policy delle risorse, quindi seleziona Crea policy.
4. Immetti la seguente policy. Replace (Sostituisci) *`vpceID`* con il tuo ID VPC endpoint ricavato dagli output del tuo AWS CloudFormation stack.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "execute-api:/*",
            "Condition": {
                "StringNotEquals": {
                    "aws:sourceVpce": "vpceID"
                }
            }
        },
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
```

```
            "Resource": "execute-api:/*"
        }
    ]
}
```

5. Seleziona Salvataggio delle modifiche.

Fase 5: Implementa il tuo API

Successivamente, distribuisci il tuo API per renderlo disponibile ai clienti nel tuo AmazonVPC.

Per distribuire un API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli il tuo API.
3. Seleziona Deploy (Implementa)API.
4. In Fase, seleziona Nuova fase.
5. In Stage name (Nome fase) immettere **`test`**.
6. (Facoltativo) In Description (Descrizione), immettere una descrizione.
7. Seleziona Deploy (Implementa).

Ora sei pronto per testare il tuoAPI.

Passo 6: Verifica che il tuo API non sia accessibile al pubblico

`curl`Utilizzalo per verificare che non puoi richiamare il tuo API dall'esterno di AmazonVPC.

Per testare il tuo API

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli il tuo API.
3. Nel pannello di navigazione principale scegli Fasi, quindi seleziona la fase test.
4. In Stage details, scegli l'icona di copia per copiare API la tua chiamata. URL URLSembrahttps://*abcdef123*.execute-api.*us-west-2*.amazonaws.com/test. L'VPCendpoint che hai creato nel passaggio 1 è DNS abilitato in modalità privata, quindi puoi utilizzare il provider URL per richiamare il tuo. API
5. Usa curl per tentare di invocare il tuo API dall'esterno del tuo. VPC

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/test
```

Curl indica che il tuo endpoint API non può essere risolto. Se ricevi una risposta diversa, torna allo Step 2 e assicurati di scegliere Privato come tipo API di endpoint che utilizzi.

```
curl: (6) Could not resolve host: abcdef123.execute-api.us-west-2.amazonaws.com/
test
```

Successivamente, ti connetti a un'EC2istanza Amazon del tuo computer VPC per richiamare il tuoAPI.

Passaggio 7: Connettiti a un'istanza nel tuo VPC e richiama il API

Successivamente, esegui il test API all'interno del tuo AmazonVPC. Per accedere al tuo account privatoAPI, ti connetti a un'EC2istanza Amazon nel tuo VPC e poi usi curl per richiamare il tuo. API Per connettersi all'istanza nel browser, si utilizza Systems Manager Session Manger.

Per testare il tuo API

1. Apri la EC2 console Amazon all'indirizzo https://console.aws.amazon.com/ec2/.
2. Seleziona Instances (Istanze).
3. Scegli l'istanza denominata private-api-tutorialche hai creato con il AWS CloudFormation modello nel passaggio 1.
4. Scegliere Connect (Connetti), quindi Session Manager (Gestore di sessione).
5. Scegliere Connect per avviare una sessione basata su browser nell'istanza.
6. Nella tua sessione di Session Manager, usa curl per richiamare il tuo. API Puoi richiamare il tuo API perché stai utilizzando un'istanza nel tuo AmazonVPC.

   ```
   curl https://abcdef123.execute-api.us-west-2.amazonaws.com/test
   ```

   Verificare di ottenere la risposta `Hello from Lambda!`.

Hai creato con successo un API account accessibile solo dall'interno del tuo Amazon VPC e poi ne hai verificato il funzionamento.

Fase 8: Pulizia

Per evitare costi non necessari, eliminare le risorse create nell'ambito di questo tutorial. I passaggi seguenti eliminano il tuo REST API e il tuo AWS CloudFormation stack.

Per eliminare un REST API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nella APIspagina, seleziona unAPI. Scegliete APIIle azioniAPI, scegliete Elimina e confermate la scelta.

Per eliminare una AWS CloudFormation pila

1. Apri la AWS CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformation.
2. Seleziona il tuo AWS CloudFormation stack.
3. Scegli Elimina e conferma la tua scelta.

Passaggi successivi: automatizza con AWS CloudFormation

Puoi automatizzare la creazione e la pulizia di tutte le AWS risorse coinvolte in questo tutorial. Per un modello AWS CloudFormation di esempio completo, consulta template.yaml.

Tutorial sull'API HTTP di Amazon API Gateway

I seguenti tutorial forniscono esercizi pratici per aiutarti a usare le API HTTP di API Gateway.

Argomenti

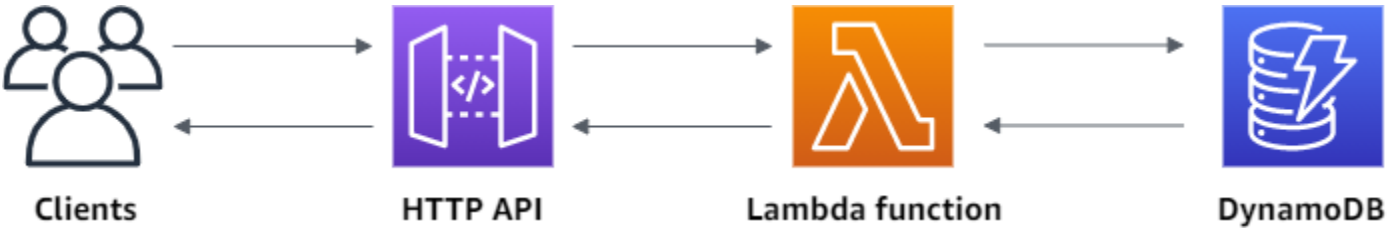
- Tutorial: Creare un'API HTTP CRUD con Lambda e DynamoDB
- Tutorial: crea un HTTP API account con un'integrazione privata con un ECS servizio Amazon

Tutorial: Creare un'API HTTP CRUD con Lambda e DynamoDB

In questo tutorial crei un'API serverless che crea, legge, aggiorna ed elimina voci da una tabella DynamoDB. DynamoDB è un servizio di database NoSQL interamente gestito che combina prestazioni elevate e prevedibili con una scalabilità continua. Questo tutorial richiede circa 30 minuti e può essere eseguito all'interno del piano gratuito di AWS.

Innanzitutto, si crea una tabella DynamoDB utilizzando la console DynamoDB. Quindi crei una funzione Lambda utilizzando la AWS Lambda console. Successivamente, è possibile creare un'API HTTP utilizzando la console API Gateway. Infine, si esegue il dell'API.

Quando si richiama l'API HTTP, API Gateway instrada la richiesta alla funzione Lambda. La funzione Lambda interagisce con DynamoDB e restituisce una risposta ad API Gateway. La console API Gateway restituisce quindi una risposta all'utente.



Per completare questo esercizio, sono necessari un AWS account e un AWS Identity and Access Management utente con accesso alla console. Per ulteriori informazioni, consulta Prerequisiti.

In questo tutorial utilizzerai AWS Management Console. Per un AWS SAM modello che crea questa API e tutte le risorse correlate, vedi `template.yaml`.

Argomenti

- Fase 1: creazione di una tabella DynamoDB
- Fase 2: creazione di una funzione Lambda
- Fase 3: creazione un'API HTTP

- Fase 4: creazione di route
- Fase 5: creazione di un'integrazione
- Fase 6: collega la tua integrazione alle route
- Fase 7: test dell'API
- Fase 8: Pulizia
- Passaggi successivi: automatizza con AWS SAM o AWS CloudFormation

Fase 1: creazione di una tabella DynamoDB

Si utilizza una tabella DynamoDB per memorizzare i dati per l'API.

Ogni voce ha un ID univoco, che usiamo come chiave di partizione per la tabella.

Per creare una tabella DynamoDB

1. Apri la console DynamoDB all'indirizzo https://console.aws.amazon.com/dynamodb/.
2. Seleziona Create table (Crea tabella).
3. Nel campo Table name (Nome tabella) immetti **`http-crud-tutorial-items`**.
4. In Partition key (Chiave di partizione), inserisci **`id`**.
5. Scegliere Create table (Crea tabella).

Fase 2: creazione di una funzione Lambda

Crei una funzione Lambda per il back-end della tua API. Questa funzione Lambda crea, legge, aggiorna ed elimina voci da DynamoDB. La funzione utilizza gli eventi da API Gateway per determinare come interagire con DynamoDB. Per semplicità questo tutorial utilizza una singola funzione Lambda. Come best practice, dovresti creare funzioni separate per ogni route.

Per creare una funzione Lambda

1. Accedere alla console Lambda all'indirizzo https://console.aws.amazon.com/lambda/.
2. Selezionare Create function (Crea funzione).
3. Nel campo Function name (Nome funzione), immettere **`http-crud-tutorial-function`**.
4. Per Runtime, scegli l'ultimo runtime supportato di Node.js o di Python.
5. In Permissions (Autorizzazioni) scegli Change default execution role (Cambia ruolo di esecuzione predefinito).

6. Seleziona Crea un nuovo ruolo dai modelli di AWS policy.
7. In Role name (Nome ruolo), immettere **`http-crud-tutorial-role`**.
8. In Policy templates (Modelli di policy) scegli **`Simple microservice permissions`**. Questa policy concede alla funzione Lambda l'autorizzazione per interagire con DynamoDB.

> **Note**
>
> Per semplicità, questo tutorial utilizza una policy gestita. Come best practice, dovresti creare la tua policy IAM per concedere le autorizzazioni minime richieste.

9. Selezionare Create function (Crea funzione).
10. Apri la funzione Lambda nell'editor di codice della console e sostituisci il suo contenuto con il codice seguente. Scegli Deploy (Distribuisci) per aggiornare la funzione.

Node.js

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  ScanCommand,
  PutCommand,
  GetCommand,
  DeleteCommand,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});

const dynamo = DynamoDBDocumentClient.from(client);

const tableName = "http-crud-tutorial-items";

export const handler = async (event, context) => {
  let body;
  let statusCode = 200;
  const headers = {
    "Content-Type": "application/json",
  };

  try {
    switch (event.routeKey) {
```

```
      case "DELETE /items/{id}":
        await dynamo.send(
          new DeleteCommand({
            TableName: tableName,
            Key: {
              id: event.pathParameters.id,
            },
          })
        );
        body = `Deleted item ${event.pathParameters.id}`;
        break;
      case "GET /items/{id}":
        body = await dynamo.send(
          new GetCommand({
            TableName: tableName,
            Key: {
              id: event.pathParameters.id,
            },
          })
        );
        body = body.Item;
        break;
      case "GET /items":
        body = await dynamo.send(
          new ScanCommand({ TableName: tableName })
        );
        body = body.Items;
        break;
      case "PUT /items":
        let requestJSON = JSON.parse(event.body);
        await dynamo.send(
          new PutCommand({
            TableName: tableName,
            Item: {
              id: requestJSON.id,
              price: requestJSON.price,
              name: requestJSON.name,
            },
          })
        );
        body = `Put item ${requestJSON.id}`;
        break;
      default:
        throw new Error(`Unsupported route: "${event.routeKey}"`);
```

```
    }
  } catch (err) {
    statusCode = 400;
    body = err.message;
  } finally {
    body = JSON.stringify(body);
  }

  return {
    statusCode,
    body,
    headers,
  };
};
```

Python

```
import json
import boto3
from decimal import Decimal

client = boto3.client('dynamodb')
dynamodb = boto3.resource("dynamodb")
table = dynamodb.Table('http-crud-tutorial-items')
tableName = 'http-crud-tutorial-items'


def lambda_handler(event, context):
    print(event)
    body = {}
    statusCode = 200
    headers = {
        "Content-Type": "application/json"
    }

    try:
        if event['routeKey'] == "DELETE /items/{id}":
            table.delete_item(
                Key={'id': event['pathParameters']['id']})
            body = 'Deleted item ' + event['pathParameters']['id']
        elif event['routeKey'] == "GET /items/{id}":
            body = table.get_item(
                Key={'id': event['pathParameters']['id']})
```

```
            body = body["Item"]
            responseBody = [
                {'price': float(body['price']), 'id': body['id'], 'name':
body['name']}]
            body = responseBody
        elif event['routeKey'] == "GET /items":
            body = table.scan()
            body = body["Items"]
            print("ITEMS----")
            print(body)
            responseBody = []
            for items in body:
                responseItems = [
                    {'price': float(items['price']), 'id': items['id'], 'name':
items['name']}]
                responseBody.append(responseItems)
            body = responseBody
        elif event['routeKey'] == "PUT /items":
            requestJSON = json.loads(event['body'])
            table.put_item(
                Item={
                    'id': requestJSON['id'],
                    'price': Decimal(str(requestJSON['price'])),
                    'name': requestJSON['name']
                })
            body = 'Put item ' + requestJSON['id']
    except KeyError:
        statusCode = 400
        body = 'Unsupported route: ' + event['routeKey']
    body = json.dumps(body)
    res = {
        "statusCode": statusCode,
        "headers": {
            "Content-Type": "application/json"
        },
        "body": body
    }
    return res
```

Fase 3: creazione un'API HTTP

L'API HTTP fornisce un endpoint HTTP per la funzione Lambda. In questa fase si crea un'API vuota. Nelle fasi seguenti si configurano route e integrazioni per connettere l'API e la funzione Lambda.

Per creare un'API HTTP

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli Create API (Crea API), quindi per API HTTP, scegli Build (Crea).
3. Per API name (Nome API), immettere **`http-crud-tutorial-api`**.
4. Seleziona Successivo.
5. Per Configura route, scegli Avanti per ignorare la creazione della route. È possibile creare route in un secondo momento.
6. Esamina la fase creata da API Gateway e quindi scegli Next (Avanti).
7. Seleziona Crea.

Fase 4: creazione di route

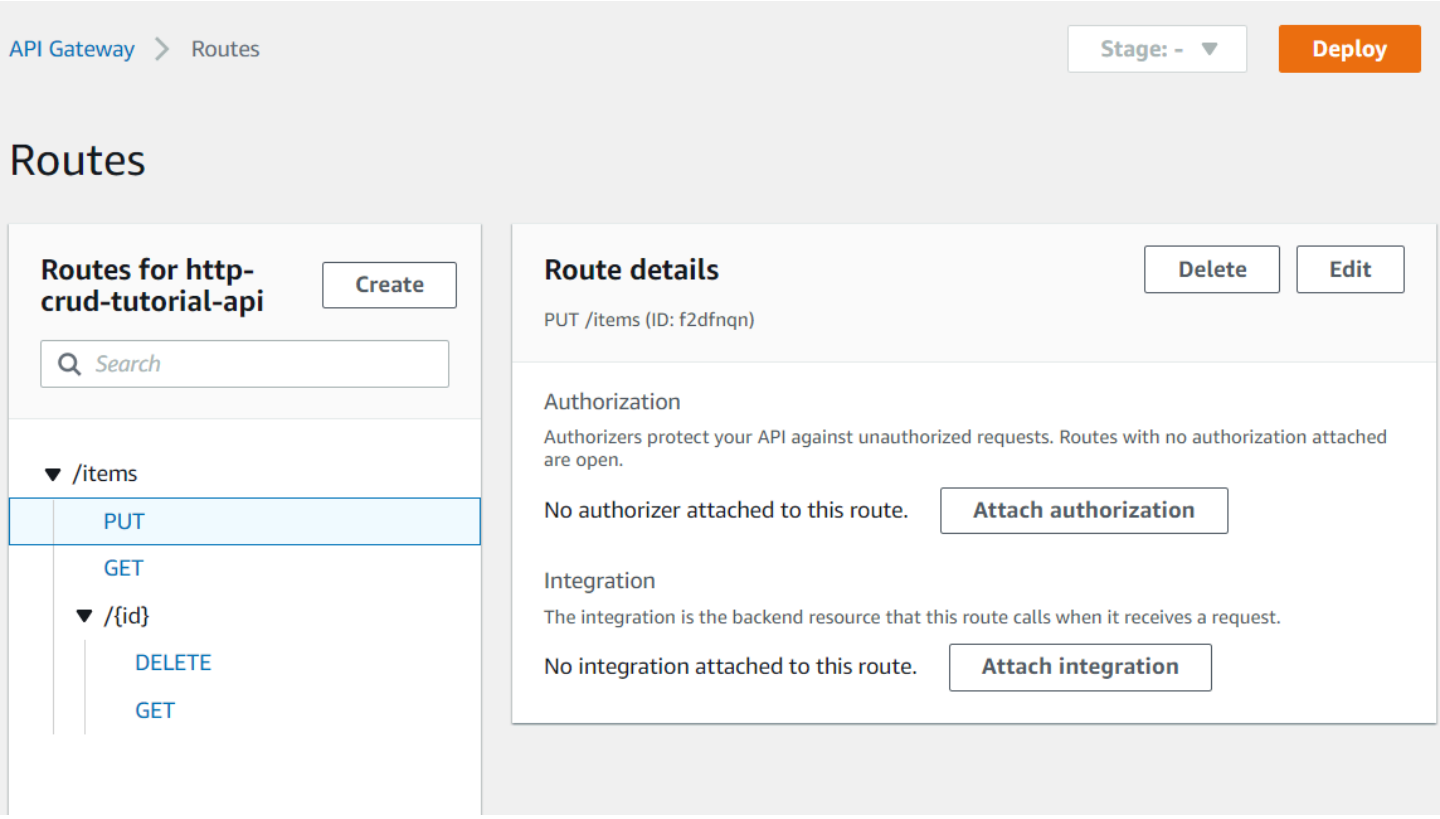
Le route inviano le richieste API in entrata alle risorse di back-end. Le route sono costituite da due parti: un metodo HTTP e un percorso della risorsa, ad esempi, `GET /items`. Per questa API di esempio, creiamo quattro route:

- `GET /items/{id}`
- `GET /items`
- `PUT /items`
- `DELETE /items/{id}`

Per creare route

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere l'API.
3. Scegliere Percorsi.
4. Seleziona Crea.
5. Per Method (Metodo) seleziona **`GET`**.

6. Per il percorso, immetti **`/items/{id}`**. Il `{id}` alla fine del percorso è un parametro che API Gateway recupera dal percorso della richiesta quando essa viene effettuata da un client.
7. Seleziona Crea.
8. Ripeti le fasi da 4 a 7 per `GET /items`, `DELETE /items/{id}` e `PUT /items`.



Fase 5: creazione di un'integrazione

Si crea un'integrazione per connettere una route alle risorse di back-end. Per questa API di esempio, si crea un'integrazione Lambda che si utilizza per tutte le route.

Per creare un'integrazione

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere l'API.
3. Scegli Integrations (Integrazioni).
4. Scegli Manage integrations (Gestisci integrazioni), quindi scegli Create (Crea).
5. Salta il passaggio Collega questa integrazione a una route. Lo completerai in una fase successiva.

6. Per Integration type (Tipo di integrazione), scegli Lambda Function (Funzione Lambda).
7. Per Lambda function (Funzione Lambda), immetti **`http-crud-tutorial-function`**.
8. Seleziona Crea.

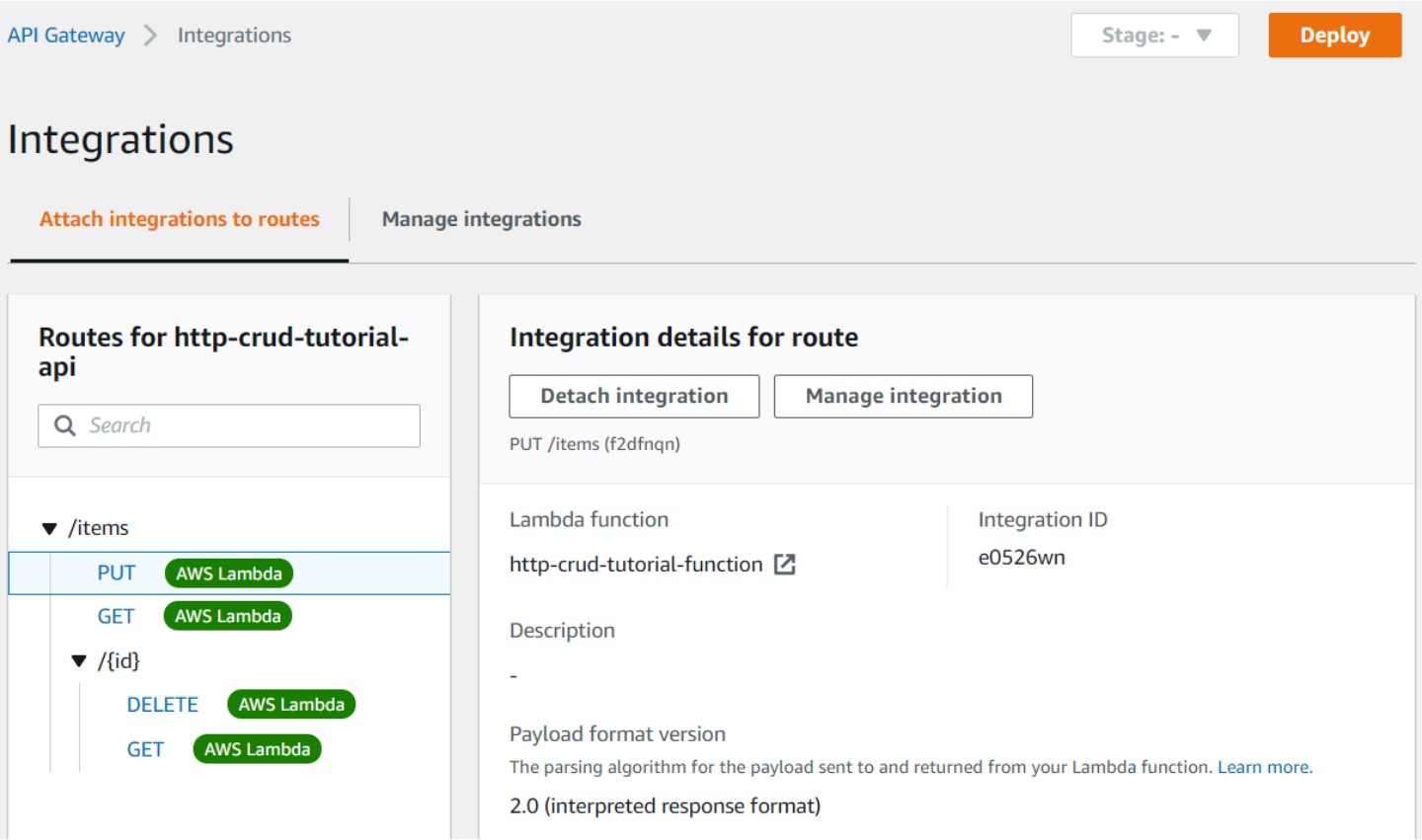
Fase 6: collega la tua integrazione alle route

Per questa API di esempio, si utilizza la stessa integrazione Lambda per tutte le route. Dopo aver collegato l'integrazione a tutte le route dell'API, la funzione Lambda viene richiamata quando un client chiama una delle route.

Per collegare integrazioni alle route

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere l'API.
3. Scegli Integrations (Integrazioni).
4. Scegli una route.
5. Nel campo Choose an existing integration (Scegli un'integrazione esistente), scegli **`http-crud-tutorial-function`**.
6. Scegli Collega integrazione.
7. Ripeti le fasi 4-6 per tutte le route.

Tutti i percorsi mostrano che è associata un' AWS Lambda integrazione.



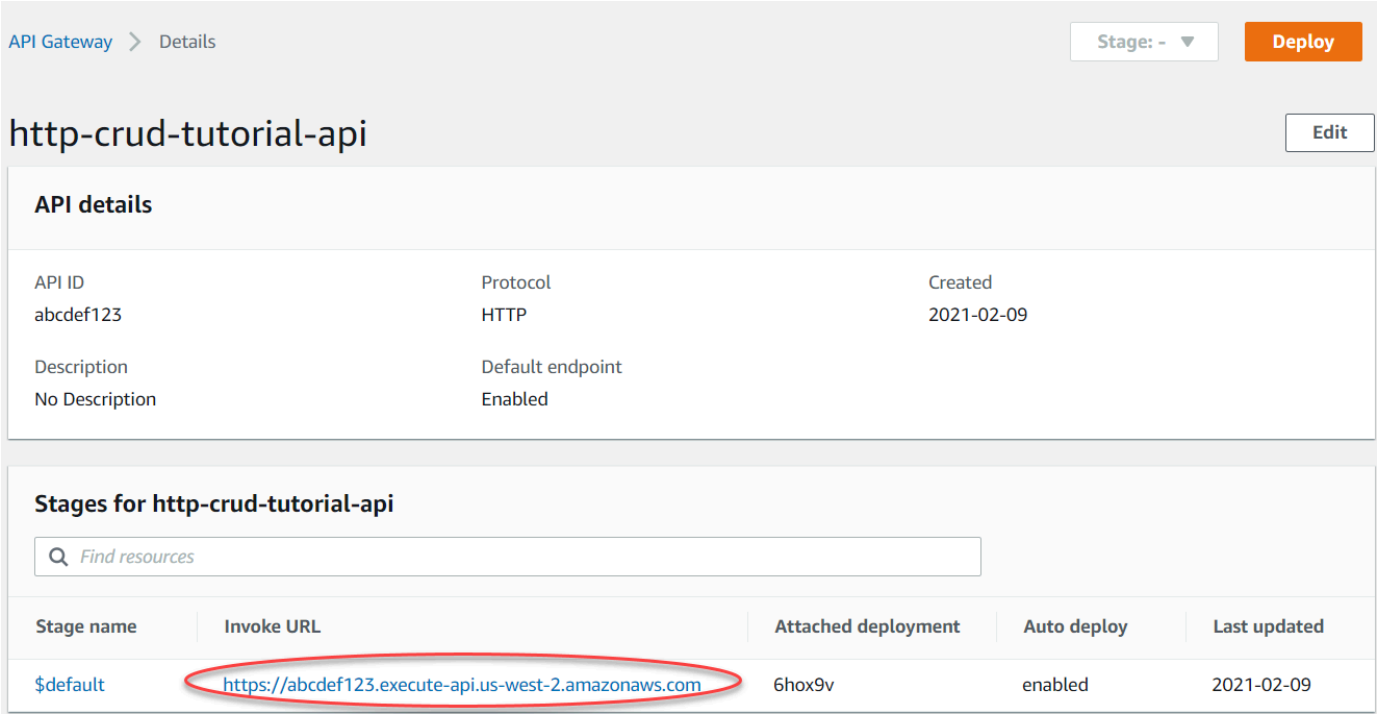
Ora che hai un'API HTTP con route e integrazioni, puoi testare la tua API.

Fase 7: test dell'API

Per assicurarti che la tua API funzioni, usa curl.

Per ottenere l'URL per richiamare la tua API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere l'API.
3. Prendere nota del valore URL di chiamata per l'API. Viene visualizzato in Invoke URL (Richiama URL) nella pagina Details (Dettagli).



4. Copia l'URL di chiamata per l'API.

   L'URL completo è simile a `https://abcdef123.execute-api.us-west-2.amazonaws.com`.

Per creare o aggiornare una voce

- Utilizza il comando seguente per creare o aggiornare una voce. Il comando include un corpo della richiesta con l'ID, il prezzo e il nome della voce.

```
curl -X "PUT" -H "Content-Type: application/json" -d "{\"id\": \"123\",
 \"price\": 12345, \"name\": \"myitem\"}" https://abcdef123.execute-api.us-
west-2.amazonaws.com/items
```

Per ottenere tutte le voci

- Utilizza il seguente comando per elencare tutte le voci.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/items
```

Per ottenere una voce

- Utilizza il seguente comando per ottenere una voce in base al relativo ID.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/items/123
```

Per eliminare una voce

1. Utilizza il comando seguente per eliminare una voce.

```
curl -X "DELETE" https://abcdef123.execute-api.us-west-2.amazonaws.com/items/123
```

2. Ottieni tutte le voci per verificare che la voce sia stata eliminata.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/items
```

Fase 8: Pulizia

Per evitare costi non necessari, eliminare le risorse create nell'ambito di questo esercizio di nozioni di base. La procedura seguente elimina l'API HTTP, la funzione Lambda e le risorse associate.

Per eliminare una tabella DynamoDB

1. Apri la console DynamoDB all'indirizzo https://console.aws.amazon.com/dynamodb/.
2. Seleziona la tabella.
3. Seleziona Delete Table (Elimina tabella).
4. Conferma la scelta e seleziona Delete (Elimina).

Per eliminare un'API HTTP

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nella pagina API, selezionare un'API. Scegli Azioni, quindi Elimina.
3. Scegliere Delete (Elimina).

Per eliminare una funzione Lambda

1. Accedere alla console Lambda all'indirizzo https://console.aws.amazon.com/lambda/.
2. Nella pagina Funzioni, selezionare una funzione. Scegli Azioni, quindi Elimina.
3. Scegliere Delete (Elimina).

Per eliminare il gruppo di log di una funzione Lambda

1. Nella CloudWatch console Amazon, apri la pagina Log groups.
2. Nella pagina Log Groups (Gruppi di log), seleziona il gruppo di log della funzione (`/aws/lambda/http-crud-tutorial-function`). Scegliere Actions (Operazioni), quindi selezionare Delete log group (Elimina gruppo di log).
3. Scegliere Delete (Elimina).

Per eliminare il ruolo di esecuzione di una funzione Lambda

1. Nella AWS Identity and Access Management console, apri la pagina Ruoli.
2. Seleziona il ruolo della funzione, ad esempi, `http-crud-tutorial-role`.
3. Scegliere Delete role (Elimina ruolo).
4. Scegliere Yes, delete (Sì, elimina).

Passaggi successivi: automatizza con AWS SAM o AWS CloudFormation

È possibile automatizzare la creazione e la pulizia delle AWS risorse utilizzando o. AWS CloudFormation AWS SAM Per un esempio di modello AWS SAM per questo tutorial, consulta `template.yaml`.

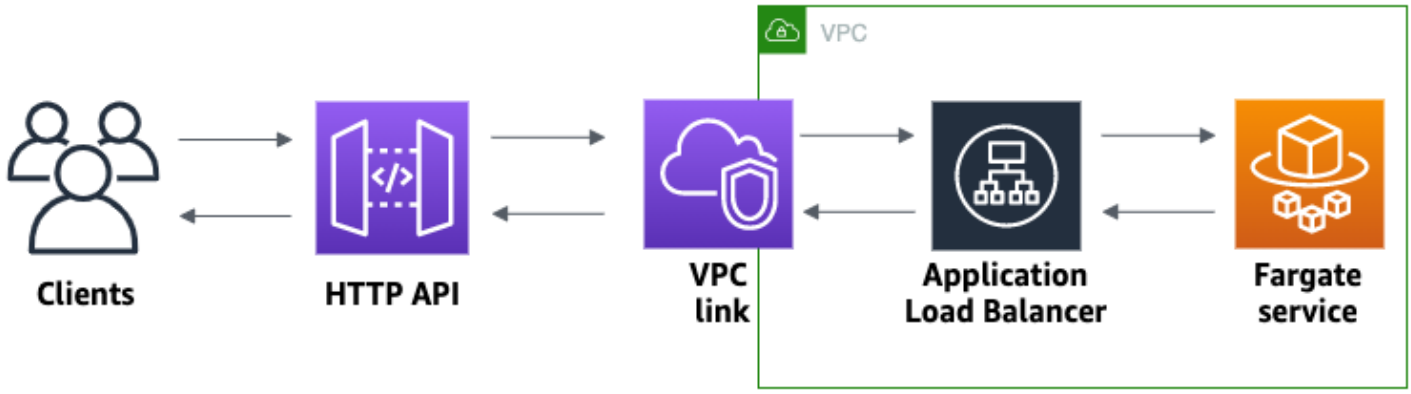
Per esempio AWS CloudFormation modelli, vedi modelli di esempio AWS CloudFormation.

Tutorial: crea un HTTP API account con un'integrazione privata con un ECS servizio Amazon

In questo tutorial, crei un serverless API che si connette a un ECS servizio Amazon eseguito su AmazonVPC. I clienti esterni al tuo Amazon VPC possono utilizzare il API per accedere al tuo ECS servizio Amazon.

Questo tutorial dura circa un'ora. Innanzitutto, utilizzi un AWS CloudFormation modello per creare un ECS servizio Amazon VPC e Amazon. Quindi usi la console API Gateway per creare un VPC collegamento. Il VPC collegamento consente a API Gateway di accedere al ECS servizio Amazon eseguito sul tuo AmazonVPC. Successivamente, crei un HTTP API account che utilizza il VPC link per connettersi al tuo ECS servizio Amazon. Infine, metti alla prova il tuoAPI.

Quando richiami il tuo HTTPAPI, API Gateway indirizza la richiesta al tuo ECS servizio Amazon tramite il tuo VPC link, quindi restituisce la risposta dal servizio.



Per completare questo tutorial, sono necessari un AWS account e un AWS Identity and Access Management utente con accesso alla console. Per ulteriori informazioni, consulta Prerequisiti.

In questo tutorial utilizzerai AWS Management Console. Per un AWS CloudFormation modello che crea questa API e tutte le risorse correlate, vedi template.yaml.

Argomenti

- Passaggio 1: creare un ECS servizio Amazon
- Passaggio 2: crea un link VPC
- Passaggio 3: Crea un HTTP API
- Fase 4: Creazione di una route
- Fase 5: creazione di un'integrazione
- Passo 6: Metti alla prova il tuo API
- Fase 7: pulire
- Passaggi successivi: automatizza con AWS CloudFormation

Passaggio 1: creare un ECS servizio Amazon

Amazon ECS è un servizio di gestione dei container che semplifica l'esecuzione, l'arresto e la gestione dei contenitori Docker su un cluster. In questo tutorial, esegui il tuo cluster su un'infrastruttura serverless gestita da AmazonECS.

Scarica e decomprimi questo AWS CloudFormation modello, che crea tutte le dipendenze per il servizio, incluso un Amazon. VPC Utilizza il modello per creare un ECS servizio Amazon che utilizza un Application Load Balancer.

Per creare uno stack AWS CloudFormation

1. Apri la AWS CloudFormation console in https://console.aws.amazon.com/cloudformation.
2. Scegliere Create stack (Crea stack), quindi With new resources (standard) (Con nuove risorse (standard)).
3. In Specificare modello, scegliere Carica un file modello.
4. Selezionare il modello scaricato.
5. Scegli Next (Successivo).
6. Per Stack name (Nome stack), inserire **`http-api-private-integrations-tutorial`**, quindi scegliere Next (Avanti).
7. Per Configure stack options (Configura opzioni di stack), scegliere Next (Successivo).
8. Per quanto riguarda le funzionalità, riconosci che AWS CloudFormation puoi creare IAM risorse nel tuo account.
9. Scegli Invia.

AWS CloudFormation fornisce il ECS servizio, operazione che può richiedere alcuni minuti. Quando lo stato del tuo AWS CloudFormation stack è CREATE_ COMPLETE, sei pronto per passare alla fase successiva.

Passaggio 2: crea un link VPC

Un VPC collegamento consente a API Gateway di accedere a risorse private in un AmazonVPC. Utilizzi un VPC link per consentire ai clienti di accedere al tuo ECS servizio Amazon tramite il tuo HTTPAPI.

Per creare un VPC link

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel riquadro di navigazione principale, scegli VPCi link, quindi scegli Crea.

   Potrebbe essere necessario scegliere l'icona del menu per aprire il pannello di navigazione principale.
3. In Scegli una versione del VPC link, seleziona il VPClink per HTTP APIs.
4. Per Nome, immetti **`private-integrations-tutorial`**.
5. Per VPC, scegli quello VPC che hai creato nel passaggio 1. Il nome dovrebbe iniziare con PrivateIntegrationsStack.
6. Per Sottoreti, seleziona le due sottoreti private nel tuo. VPC I nomi finiscono con `PrivateSubnet`.
7. Per i gruppi di sicurezza, seleziona l'ID del gruppo che inizia con `private-integrations-tutorial` e contiene la descrizione di. `PrivateIntegrationsStack/PrivateIntegrationsTutorialService/Service/SecurityGroup`
8. Scegli Create (Crea) .

Dopo aver creato il VPC collegamento, API Gateway fornisce interfacce di rete elastiche per accedere VPC a. Il processo può richiedere alcuni minuti. Nel frattempo, puoi creare il tuo. API

Passaggio 3: Crea un HTTP API

HTTPAPIFornisce un HTTP endpoint per il tuo ECS servizio Amazon. In questo passaggio, crei un vuotoAPI. Nei passaggi 4 e 5, configuri un percorso e un'integrazione per connettere il tuo API e il tuo ECS servizio Amazon.

Per creare un HTTP API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliete Crea API, quindi per HTTPAPI, scegliete Costruisci.
3. Per APInome, inserisci**`http-private-integrations-tutorial`**.
4. Scegli Next (Successivo).
5. Per Configura route, scegli Avanti per ignorare la creazione della route. È possibile creare route in un secondo momento.

6. Rivedi la fase che API Gateway crea per te. APIGateway crea una `$default` fase con le implementazioni automatiche abilitate, che è la scelta migliore per questo tutorial. Scegli Next (Successivo).
7. Seleziona Crea.

Fase 4: Creazione di una route

I percorsi sono un modo per inviare API richieste in entrata alle risorse di backend. I percorsi sono costituiti da due parti: un HTTP metodo e un percorso di risorse, ad esempio. `GET /items` Per questo esempioAPI, creiamo un percorso.

Per creare una route

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. APIScegli il tuo.
3. Scegliere Percorsi.
4. Seleziona Crea.
5. Per Method (Metodo) seleziona **`ANY`**.
6. Per il percorso, immetti **`/{proxy+}`**. Il`{proxy+}` alla fine del percorso è una variabile di percorso greedy. APIGateway invia tutte le richieste API a te su questa rotta.
7. Scegli Create (Crea) .

Fase 5: creazione di un'integrazione

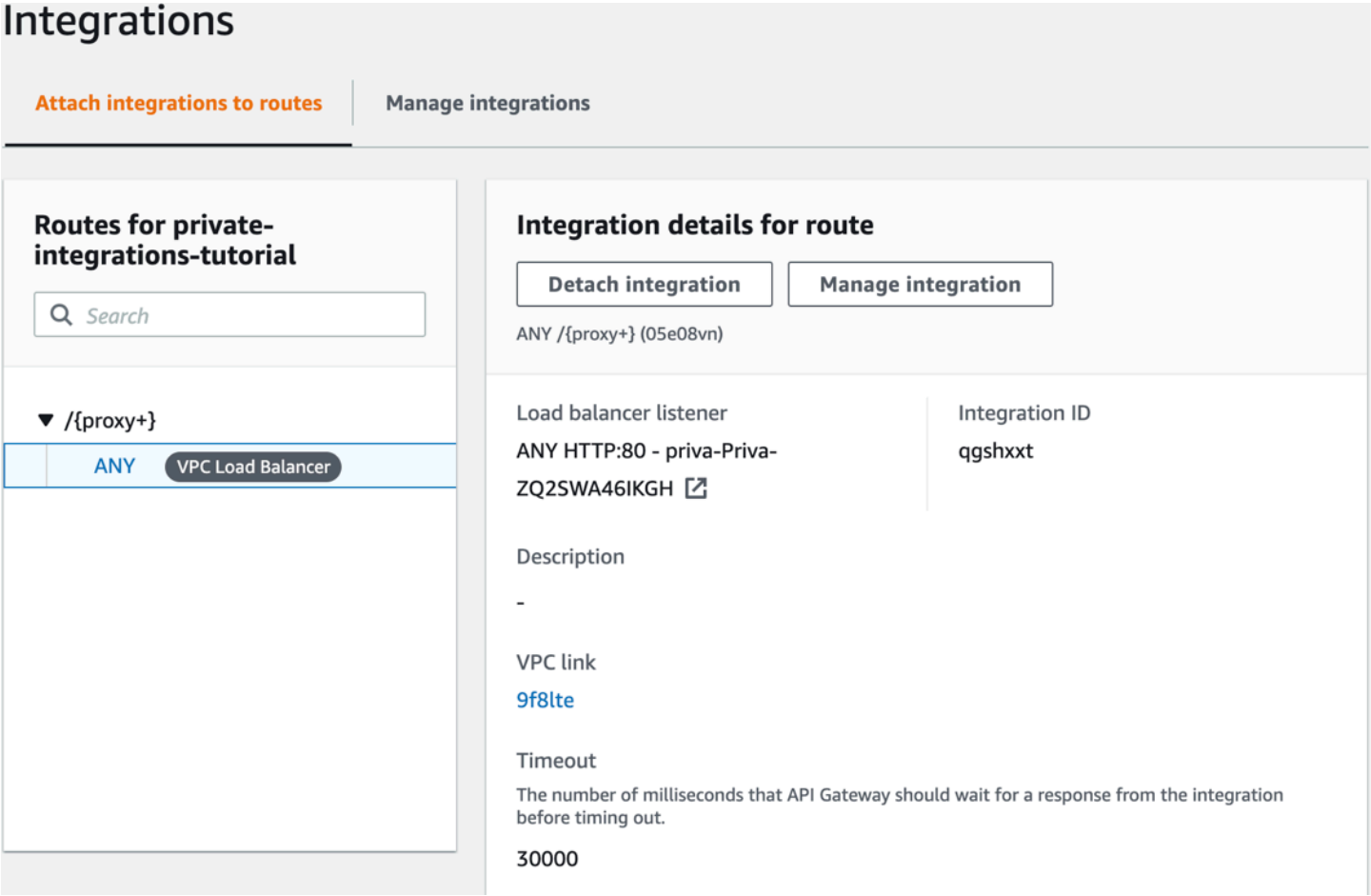
Si crea un'integrazione per connettere una route alle risorse di back-end.

Per creare un'integrazione

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. APIScegli il tuo.
3. Scegli Integrations (Integrazioni).
4. Scegli Manage integrations (Gestisci integrazioni), quindi scegli Create (Crea).
5. Per Allega questa integrazione a un percorso, seleziona il percorso ANY/{proxy+} che hai creato in precedenza.
6. Per Tipo di integrazionescegliere Risorsa privata.

7. Per Dettagli sull'integrazione, scegliere Seleziona manualmente.
8. Per il servizio Target, scegliALB/. NLB
9. Per load balancer, scegliere il load balancer creato con il modello AWS CloudFormation nella fase 1. Il suo nome dovrebbe iniziare con HTTP-Priva.
10. Per Listener, scegliere **`HTTP 80`**.
11. Per il VPCcollegamento, scegli il VPC link che hai creato nel passaggio 2. Dovrebbe chiamarsi `private-integrations-tutorial`.
12. Seleziona Crea.

Per verificare che il percorso e l'integrazione siano configurati correttamente, selezionare Allega integrazioni alle route. La console mostra che hai un `ANY /{proxy+}` percorso con un'integrazione con un Load VPC Balancer.



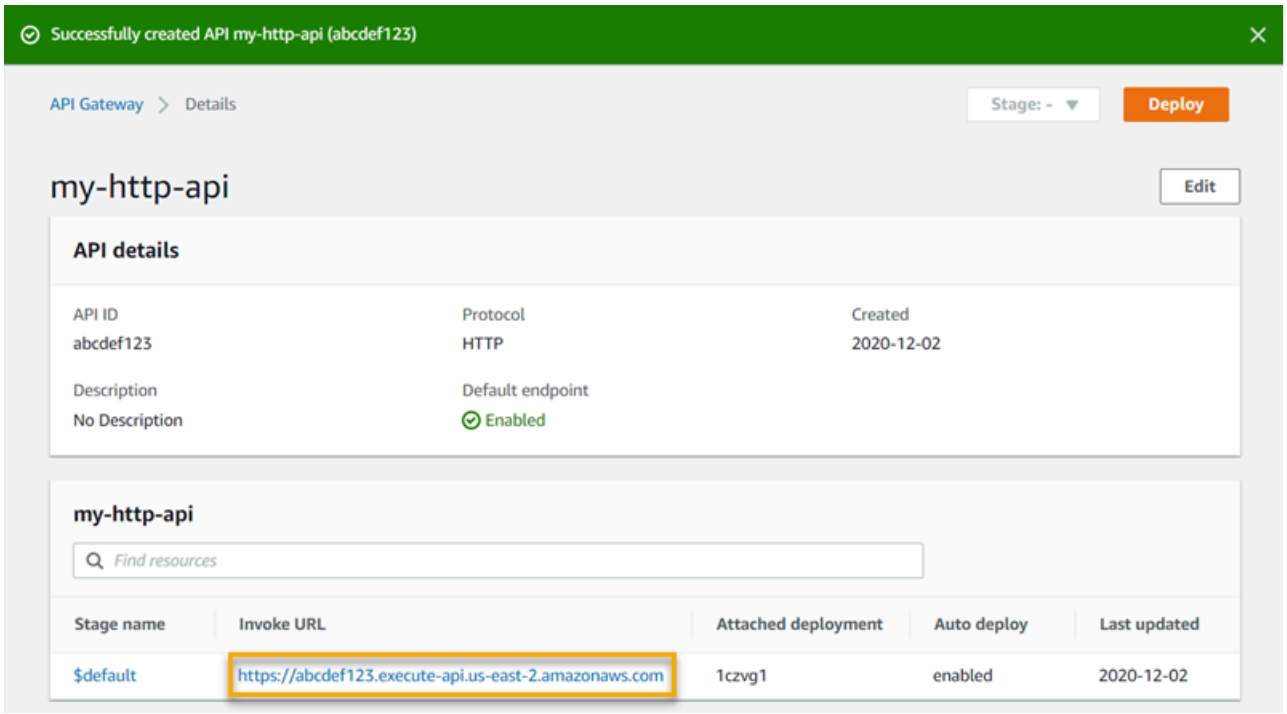
Ora sei pronto per testare il tuoAPI.

Passo 6: Metti alla prova il tuo API

Successivamente, testate il vostro API per assicurarvi che funzioni. Per semplicità, usa un browser web per richiamare il tuoAPI.

Per testare il tuo API

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. APIScegli il tuo.
3. Annota la tua API URL invocazione.



4. In un browser web, vai alla tua API URL invoke.

   La versione completa URL dovrebbe `https://abcdef123.execute-api.us-east-2.amazonaws.com` assomigliare a.

   Il tuo browser invia una `GET` richiesta aAPI.

5. Verifica che la tua risposta API sia un messaggio di benvenuto che ti informa che la tua app è in esecuzione su AmazonECS.

   Se vedi il messaggio di benvenuto, significa che hai creato con successo un ECS servizio Amazon che funziona su Amazon VPC e hai utilizzato un API Gateway HTTP API con un VPC link per accedere al ECS servizio Amazon.

Fase 7: pulire

Per evitare costi non necessari, eliminare le risorse create nell'ambito di questo tutorial. I passaggi seguenti eliminano il VPC link, AWS CloudFormation lo stack e HTTPAPI.

Per eliminare un HTTP API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nella APIspagina, seleziona un. API Scegliere Azioni, scegliere Elimina, quindi confermare la scelta.

Per eliminare un VPC link

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli il link. VPC
3. Seleziona il VPC link, scegli Elimina, quindi conferma la scelta.

Per eliminare una AWS CloudFormation pila

1. Apri la AWS CloudFormation console in https://console.aws.amazon.com/cloudformation.
2. Seleziona il tuo stack. AWS CloudFormation
3. Scegli Elimina e conferma la tua scelta.

Passaggi successivi: automatizza con AWS CloudFormation

Puoi automatizzare la creazione e la pulizia di tutte le AWS risorse coinvolte in questo tutorial. Per un modello AWS CloudFormation di esempio completo, consulta template.yaml.

Tutorial sulle API di Amazon WebSocket API Gateway

I seguenti tutorial forniscono un esercizio pratico per aiutarti a conoscere le API API Gateway. WebSocket

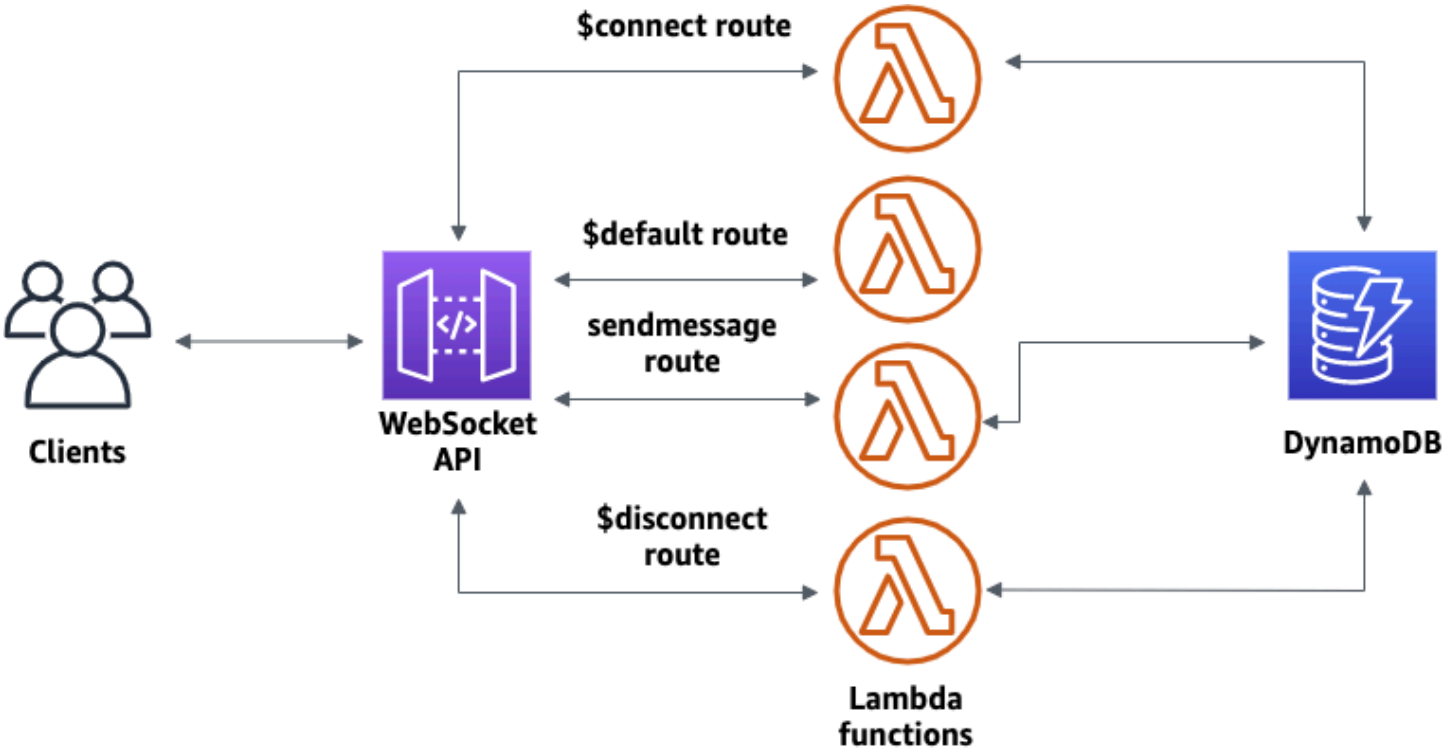
Argomenti

- Tutorial: crea un'app di WebSocket chat con a WebSocket API, Lambda e DynamoDB
- Tutorial: Crea un WebSocket API file con un' AWS integrazione

Tutorial: crea un'app di WebSocket chat con a WebSocket API, Lambda e DynamoDB

In questo tutorial, creerai un'applicazione di chat senza server con un. WebSocket API Con a WebSocket API, puoi supportare la comunicazione bidirezionale tra i client. I client possono ricevere messaggi senza dover eseguire il polling per gli aggiornamenti.

Il completamento di questa esercitazione richiede circa 30 minuti. Innanzitutto, utilizzerai un AWS CloudFormation modello per creare funzioni Lambda che gestiranno le API richieste, oltre a una tabella DynamoDB che memorizza il tuo client. IDs Quindi, utilizzerai la console API Gateway per crearne una WebSocket API che si integri con le tue funzioni Lambda. Infine, eseguirai un test API per verificare che i messaggi vengano inviati e ricevuti.



Per completare questo tutorial, sono necessari un AWS account e un AWS Identity and Access Management utente con accesso alla console. Per ulteriori informazioni, consulta Prerequisiti.

Devi anche `wscat` connetterti al tuoAPI. Per ulteriori informazioni, consulta the section called "Utilizzalo `wscat` per connetterti a un' WebSocket API e inviarle messaggi".

Argomenti

- Fase 1: creazione di funzioni Lambda e di una tabella DynamoDB
- Fase 2: Creare un WebSocket API

- Fase 3: Metti alla prova il tuo API
- Fase 4: pulizia
- Passaggi successivi: automatizza con AWS CloudFormation

Fase 1: creazione di funzioni Lambda e di una tabella DynamoDB

Scarica e decomprimi il modello di creazione dell'app per AWS CloudFormation. Utilizzerai questo modello per creare una tabella Amazon DynamoDB per archiviare il client della tua app. IDs Ogni client connesso ha un ID univoco che utilizzeremo come chiave di partizione della tabella. Questo modello crea anche funzioni Lambda che aggiornano le connessioni client in DynamoDB e gestiscono l'invio di messaggi ai client connessi.

Per creare uno stack AWS CloudFormation

1. Apri la AWS CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformation.
2. Scegliere Create stack (Crea stack), quindi With new resources (standard) (Con nuove risorse (standard)).
3. In Specificare modello, scegliere Carica un file modello.
4. Selezionare il modello scaricato.
5. Seleziona Successivo.
6. Per Stack name (Nome stack), inserire **`websocket-api-chat-app-tutorial`**, quindi scegliere Next (Avanti).
7. Per Configure stack options (Configura opzioni di stack), scegliere Next (Successivo).
8. Per quanto riguarda le funzionalità, riconosci che AWS CloudFormation puoi creare IAM risorse nel tuo account.
9. Scegli Invia.

AWS CloudFormation fornisce le risorse specificate nel modello. Per completare il provisioning delle risorse, potrebbero essere necessari alcuni minuti. Quando lo stato del tuo AWS CloudFormation stack è CREATE_ COMPLETE, sei pronto per passare alla fase successiva.

Fase 2: Creare un WebSocket API

Creerai un file WebSocket API per gestire le connessioni client e indirizzare le richieste alle funzioni Lambda che hai creato nel passaggio 1.

Per creare un WebSocket API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli Crea API. Quindi WebSocket API, scegli Costruisci.
3. Per APInome, inserisci**`websocket-chat-app-tutorial`**.
4. Per Route selection expression (Espressione di selezione routing), inserire **`request.body.action`**. L'espressione di selezione del percorso determina il percorso che API Gateway richiama quando un client invia un messaggio.
5. Seleziona Successivo.
6. Per Predefined routes (Routing predefiniti), scegliere Add $connect (Aggiungi $connect), Add $disconnect (Aggiungi $disconnect) e Add $default (Aggiungi $default). Le route $connect e $disconnect sono route speciali che API Gateway richiama automaticamente quando un client si connette o si disconnette da un. API APIIl gateway richiama la route quando nessun'altra `$default` route corrisponde a una richiesta.
7. Per Custom routes (Routing personalizzati), scegli Add custom route (Aggiungi routing personalizzato). Per Route key (Chiave routing), inserire **`sendmessage`**. Questo routing personalizzato gestisce i messaggi inviati ai client connessi.
8. Seleziona Successivo.
9. In Attach integrations (Collega integrazioni), per ogni routing eIntegration type (Tipo di integrazione), scegliere Lambda.

   Per Lambda, scegli la funzione Lambda corrispondente che hai creato nel passaggio 1. AWS CloudFormation Il nome di ciascuna funzione corrisponde a un routing. Ad esempio, per il routing $connect, scegliere la funzione denominata **`websocket-chat-app-tutorial-ConnectHandler`**.
10. Rivedi la fase che API Gateway crea per te. Per impostazione predefinita, API Gateway crea un nome di fase `production` e lo distribuisce automaticamente API in quella fase. Seleziona Successivo.
11. Scegliere Create and deploy (Crea e implementa).

Fase 3: Metti alla prova il tuo API

Successivamente, testerai il tuo API per assicurarti che funzioni correttamente. Usa il `wscat` comando per connetterti aAPI.

Per ottenere l'invoce URL per il tuo API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli il tuo API.
3. Scegli Stages (Fasi), quindi scegli production (produzione).
4. Annota API i tuoi WebSocket URL. URLDovrebbero assomigliare `awss://abcdef123.execute-api.us-east-2.amazonaws.com/production`.

Per connetterti al tuo API

1. Usa il seguente comando per connetterti al tuoAPI. Quando ti connetti al tuoAPI, API Gateway richiama il `$connect` percorso. Quando si richiama il routing, esso richiama una funzione Lambda che archivia l'ID di connessione in DynamoDB.

```
wscat -c wss://abcdef123.execute-api.us-west-2.amazonaws.com/production
```

```
Connected (press CTRL+C to quit)
```

2. Apri un nuovo terminale ed esegui nuovamente il comando wscat con i parametri seguenti.

```
wscat -c wss://abcdef123.execute-api.us-west-2.amazonaws.com/production
```

```
Connected (press CTRL+C to quit)
```

Ciò fornisce due client connessi in grado di scambiare messaggi.

Per inviare un messaggio

- APIGateway determina il percorso da richiamare in base all'espressione di selezione API del percorso dell'utente. L'espressione API di selezione del percorso dell'utente è`$request.body.action`. Di conseguenza, API Gateway richiama il `sendmessage` percorso quando si invia il seguente messaggio:

```
{"action": "sendmessage", "message": "hello, everyone!"}
```

La funzione Lambda associata alla route richiamata raccoglie il client da DynamoDB. IDs Quindi, la funzione chiama il API Gateway Management API e invia il messaggio a tali client. Tutti i client connessi ricevono il seguente messaggio:

```
< hello, everyone!
```

Per richiamare la tua route API $default

- APIGateway richiama la route predefinita API dell'utente quando un client invia un messaggio che non corrisponde ai percorsi definiti. La funzione Lambda associata alla `$default` route utilizza API Gateway Management API per inviare al client informazioni sulla sua connessione.

```
test
```

```
Use the sendmessage route to send a message. Your info:
 {"ConnectedAt":"2022-01-25T18:50:04.673Z","Identity":
{"SourceIp":"192.0.2.1","UserAgent":null},"LastActiveAt":"2022-01-25T18:50:07.642Z","connec
```

Per disconnetterti dal tuo API

- Premi **`CTRL+C`** per disconnetterti dal tuo. API Quando un client si disconnette dal tuoAPI, API Gateway richiama il tuo percorso. API `$disconnect` L'integrazione Lambda per il tuo `$disconnect` percorso rimuove l'ID API di connessione da DynamoDB.

Fase 4: pulizia

Per evitare costi non necessari, eliminare le risorse create nell'ambito di questo tutorial. I passaggi seguenti eliminano lo stack e. AWS CloudFormation WebSocket API

Per eliminare un WebSocket API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nella APIspagina, seleziona il tuo `websocket-chat-app-tutorial`API. Scegliere Azioni, scegliere Elimina, quindi confermare la scelta.

Per eliminare una AWS CloudFormation pila

1. Apri la AWS CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformation.
2. Seleziona il tuo AWS CloudFormation stack.
3. Scegli Elimina e conferma la tua scelta.

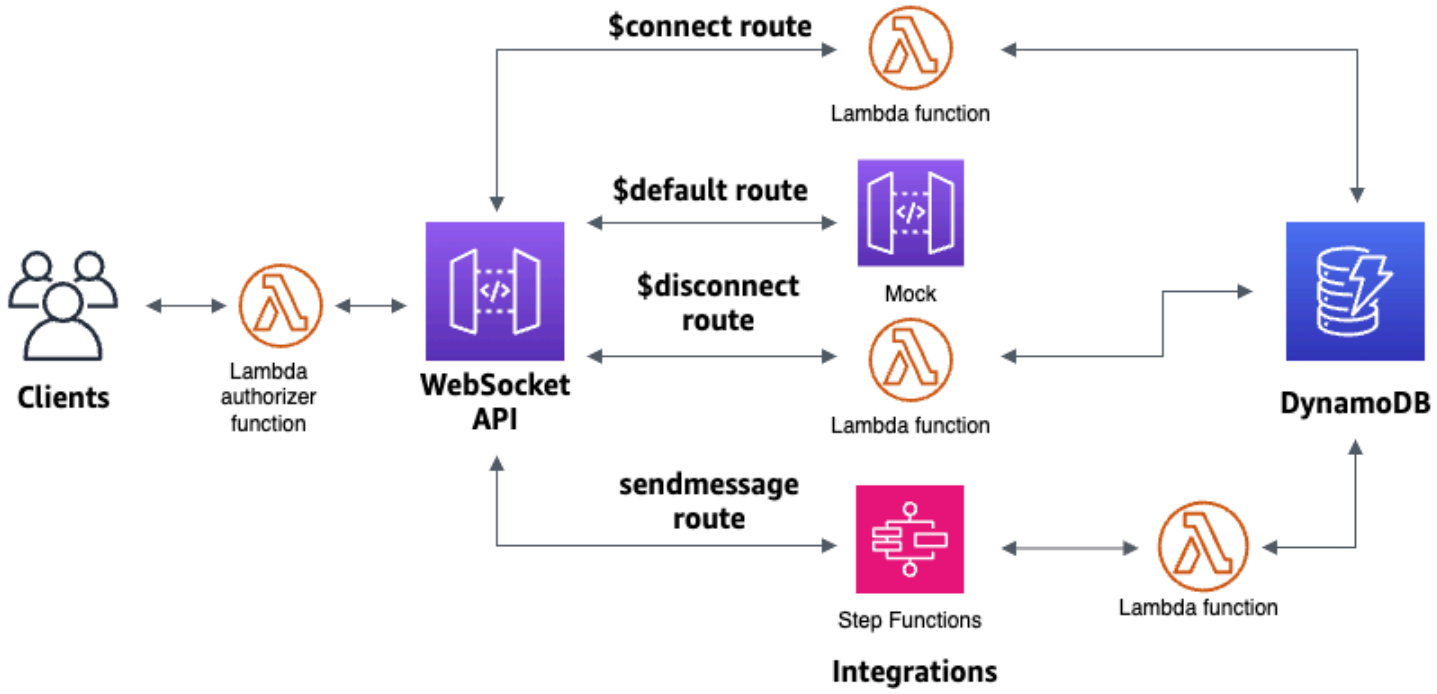
Passaggi successivi: automatizza con AWS CloudFormation

Puoi automatizzare la creazione e la pulizia di tutte le AWS risorse coinvolte in questo tutorial. Per un AWS CloudFormation modello che crea questa API e tutte le risorse correlate, vedi ws-chat-app .yaml.

Tutorial: Crea un WebSocket API file con un' AWS integrazione

In questo tutorial, crei un'applicazione di trasmissione serverless con un. WebSocket API I client possono ricevere messaggi senza dover eseguire il polling per gli aggiornamenti.

Questo tutorial mostra come trasmettere messaggi ai client connessi e include un esempio di autorizzatore Lambda, un'integrazione fittizia e un'integrazione non proxy con Step Functions.



Dopo aver creato le risorse utilizzando un AWS CloudFormation modello, utilizzerai la console API Gateway per crearne una WebSocket API che si integri con AWS le tue risorse. Collegherai un

autorizzatore Lambda al tuo API e creerai un'integrazione del AWS servizio con Step Functions per avviare l'esecuzione di una macchina a stati. La macchina a stati Step Functions invocherà una funzione Lambda che invia un messaggio a tutti i client connessi.

Dopo aver creato il tuoAPI, testerai la tua connessione al tuo API e verificherai che i messaggi vengano inviati e ricevuti. Il completamento di questo tutorial richiede circa 45 minuti.

Argomenti

- Prerequisiti
- Fase 1: Creazione delle risorse
- Passaggio 2: crea un WebSocket API
- Fase 3: Creare un autorizzatore Lambda
- Passaggio 4: Crea una finta integrazione bidirezionale
- Fase 5: Creare un'integrazione non proxy con Step Functions
- Fase 6: Metti alla prova il tuo API
- Fase 7: pulire
- Passaggi successivi

Prerequisiti

Sono necessari i seguenti prerequisiti:

- Un AWS account e un AWS Identity and Access Management utente con accesso alla console. Per ulteriori informazioni, consulta Prerequisiti.
- `wscat`per connetterti al tuoAPI. Per ulteriori informazioni, consulta the section called “Utilizzalo `wscat` per connetterti a un' WebSocket API e inviarle messaggi”.

Ti consigliamo di completare il tutorial dell'app di WebSocket chat prima di iniziare questo tutorial. Per completare il tutorial sull'app di WebSocket chat, consultathe section called “Tutorial: crea un'app di chat WebSocket ”.

Fase 1: Creazione delle risorse

Scarica e decomprimi il modello di creazione dell'app per AWS CloudFormation. Utilizzerai questo modello per creare quanto segue:

- Funzioni Lambda che gestiscono le API richieste e autorizzano l'accesso al tuo. API
- Una tabella DynamoDB per memorizzare l'identificazione IDs del client e dell'utente principale restituita dall'autorizzatore Lambda.
- Una macchina a stati Step Functions per inviare messaggi ai client connessi.

Per creare uno AWS CloudFormation stack

1. Apri la AWS CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformation.
2. Scegliere Create stack (Crea stack), quindi With new resources (standard) (Con nuove risorse (standard)).
3. In Specificare modello, scegliere Carica un file modello.
4. Selezionare il modello scaricato.
5. Seleziona Successivo.
6. Per Stack name (Nome stack), inserire **`websocket-step-functions-tutorial`**, quindi scegliere Next (Avanti).
7. Per Configure stack options (Configura opzioni di stack), scegliere Next (Successivo).
8. Per quanto riguarda le funzionalità, riconosci che AWS CloudFormation puoi creare IAM risorse nel tuo account.
9. Scegli Invia.

AWS CloudFormation fornisce le risorse specificate nel modello. Per completare il provisioning delle risorse, potrebbero essere necessari alcuni minuti. Scegli la scheda Output per vedere le risorse create e le relative. ARNs Quando lo stato del tuo AWS CloudFormation stack è CREATE_COMPLETE, sei pronto per passare alla fase successiva.

Passaggio 2: crea un WebSocket API

Creerai un file WebSocket API per gestire le connessioni dei client e indirizzare le richieste alle risorse che hai creato nel passaggio 1.

Per creare un WebSocket API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli Crea API. Quindi WebSocket API, scegli Costruisci.

3. Per APInome, inserisci**`websocket-step-functions-tutorial`**.
4. Per Route selection expression (Espressione di selezione routing), inserire **`request.body.action`**.

   L'espressione di selezione del percorso determina il percorso che API Gateway richiama quando un client invia un messaggio.
5. Seleziona Successivo.
6. Per percorsi predefiniti, scegli Aggiungi $connect, Aggiungi $disconnect, Aggiungi $default.

   Le route $connect e $disconnect sono rotte speciali che API Gateway richiama automaticamente quando un client si connette o si disconnette da un. API APIGateway richiama la route $default quando nessun'altra route corrisponde a una richiesta. Creerai un percorso personalizzato per connetterti a Step Functions dopo aver creato il tuoAPI.
7. Seleziona Successivo.
8. Per Integration for $connect, procedi come segue:

   a. Per il tipo di integrazione, scegli Lambda.

   b. Per la funzione Lambda, scegli la funzione $connect Lambda corrispondente che hai creato AWS CloudFormation nel passaggio 1. Il nome della funzione Lambda deve iniziare con. **`websocket-step`**
9. Per Integration for $disconnect, procedi come segue:

   a. Per il tipo di integrazione, scegli Lambda.

   b. Per la funzione Lambda, scegli la funzione Lambda $disconnect corrispondente che hai creato nel passaggio 1. AWS CloudFormation Il nome della funzione Lambda deve iniziare con. **`websocket-step`**
10. Per Integration for $default, scegli mock.

    In un'integrazione fittizia, API Gateway gestisce la risposta al percorso senza un backend di integrazione.
11. Seleziona Successivo.
12. Rivedi la fase che API Gateway crea per te. Per impostazione predefinita, API Gateway crea una fase denominata produzione e la distribuisce automaticamente API in quella fase. Seleziona Successivo.
13. Scegliere Create and deploy (Crea e implementa).

Fase 3: Creare un autorizzatore Lambda

Per controllare l'accesso al tuo WebSocket API, crei un autorizzatore Lambda. Il AWS CloudFormation modello ha creato la funzione di autorizzazione Lambda per te. Puoi vedere la funzione Lambda nella console Lambda. Il nome deve iniziare con. **`websocket-step-functions-tutorial-AuthorizerHandler`** Questa funzione Lambda nega tutte le chiamate a WebSocket API meno che l'`Authorization`intestazione non sia. `Allow` La funzione Lambda passa anche la `$context.authorizer.principalId` variabile al tuoAPI, che viene poi utilizzata nella tabella DynamoDB per identificare i chiamanti. API

In questo passaggio, configuri la route $connect per utilizzare l'autorizzatore Lambda.

Per creare un autorizzatore Lambda

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel riquadro di navigazione principale, scegli Autorizzazioni.
3. Scegli Crea un autorizzatore.
4. Per il nome dell'autorizzatore, inserisci. **`LambdaAuthorizer`**
5. Per Autorizzatore ARN, inserisci il nome dell'autorizzatore creato dal modello. AWS CloudFormation Il nome deve iniziare con. **`websocket-step-functions-tutorial-AuthorizerHandler`**

   > **Note**
   >
   > Ti consigliamo di non utilizzare questo autorizzatore di esempio per la tua produzioneAPIs.

6. Per il tipo di origine dell'identità, scegli Intestazione. In Chiave, inserire **`Authorization`**.
7. Scegli Create Authorizer (Crea autorizzazioni).

Dopo aver creato il tuo autorizzatore, lo colleghi alla route $connect del tuo. API

Per collegare un autorizzatore alla rotta $connect

1. Nel pannello di navigazione principale, scegli Percorsi.
2. Scegli il percorso $connect.
3. Nella sezione Impostazioni della richiesta di instradamento scegli Modifica.

4. Per Autorizzazione, scegli il menu a discesa, quindi seleziona l'autorizzatore della richiesta.
5. Seleziona Salvataggio delle modifiche.

Passaggio 4: Crea una finta integrazione bidirezionale

Successivamente, crei l'integrazione fittizia bidirezionale per il percorso $default. Un'integrazione fittizia consente di inviare una risposta al client senza utilizzare un backend. Quando crei un'integrazione per il percorso $default, puoi mostrare ai clienti come interagire con il tuo. API

Si configura la route $default per informare i client di utilizzare la route sendmessage.

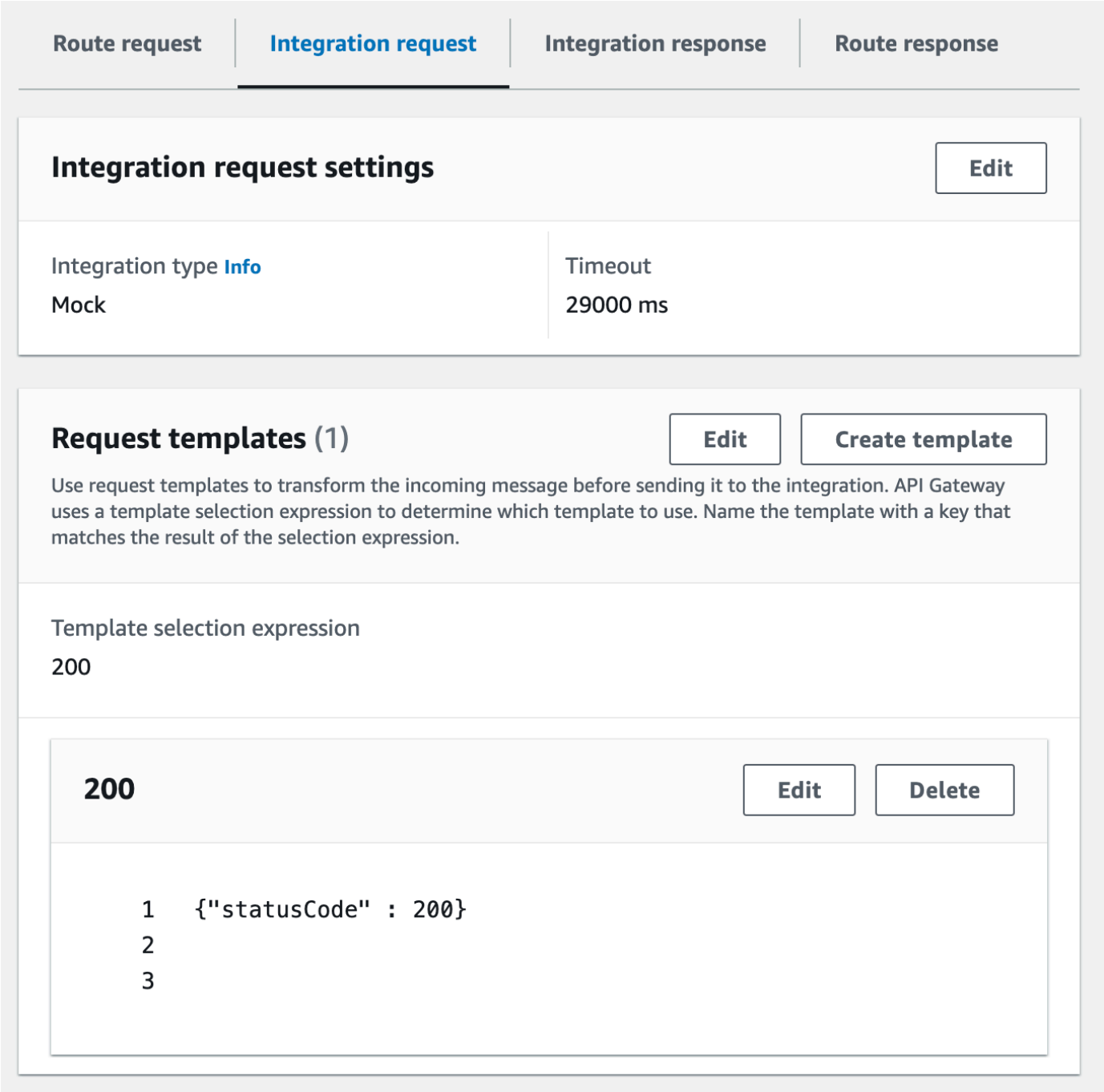
Per creare una finta integrazione

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli il percorso $default, quindi scegli la scheda Richiesta di integrazione.
3. Per i modelli di richiesta, scegli Modifica.
4. Per Espressione di selezione del modello**200**, immettete, quindi scegliete Modifica.
5. Nella scheda Richiesta di integrazione, per Richiedi modelli, scegli Crea modello.
6. Per la chiave Template, inserisci**200**.
7. Per Genera modello, inserisci il seguente modello di mappatura:

   ```
   {"statusCode": 200}
   ```

   Scegli Crea modello.

   Il risultato sarà simile al seguente:



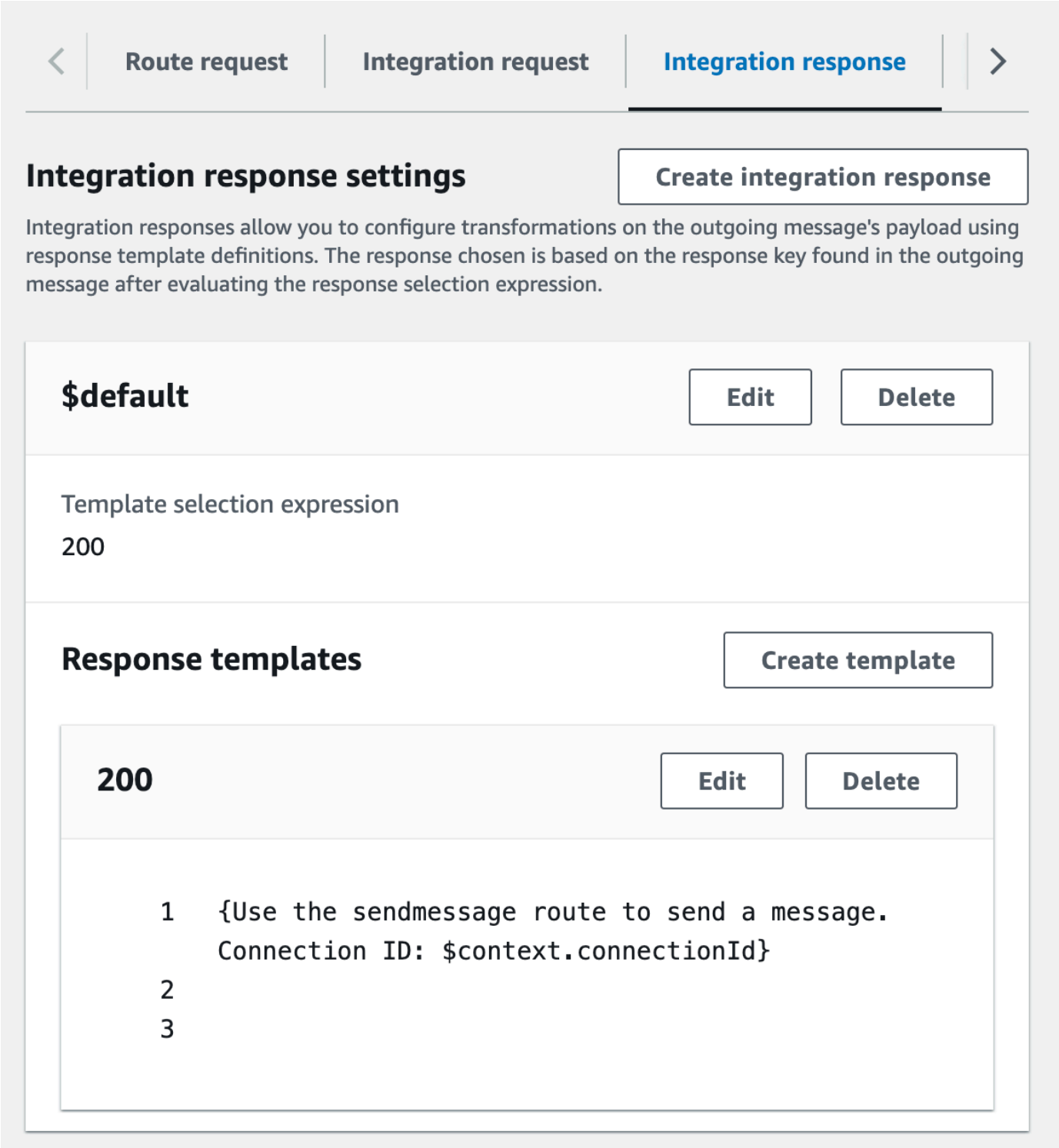
8. Nel riquadro $default route, scegli Abilita comunicazione bidirezionale.
9. Scegli la scheda Integration response, quindi scegli Crea risposta di integrazione.
10. Per la chiave di risposta, inserisci**`$default`**.
11. Per Espressione di selezione del modello, immettere**`200`**.
12. Scegli Crea risposta.
13. In Modelli di risposta, scegli Crea modello.

14. Per la chiave Template, inserisci**`200`**.
15. Per il modello di risposta, inserisci il seguente modello di mappatura:

```
{"Use the sendmessage route to send a message. Connection ID:
 $context.connectionId"}
```

16. Scegli Crea modello.

    Il risultato sarà simile al seguente:



Fase 5: Creare un'integrazione non proxy con Step Functions

Successivamente, crei un percorso di invio dei messaggi. I client possono richiamare il percorso sendmessage per trasmettere un messaggio a tutti i client connessi. Il percorso sendmessage ha

un'integrazione del servizio non proxy con. AWS AWS Step Functions L'integrazione richiama il StartExecutioncomando per la macchina a stati Step Functions che il AWS CloudFormation modello ha creato per te.

Per creare un'integrazione non proxy

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Selezionare Create Route (Crea route).
3. Per Route key (Chiave routing), inserire **`sendmessage`**.
4. Per il tipo di integrazione, scegli AWS il servizio.
5. Per AWS Regione, inserisci la regione in cui hai distribuito il AWS CloudFormation modello.
6. Per l'AWS assistenza, scegli Step Functions.
7. Per HTTPmetodo, scegli POST.
8. Per Nome azione immetti **`StartExecution`**.
9. Per Ruolo di esecuzione, immettete il ruolo di esecuzione creato dal AWS CloudFormation modello. Il nome deve essere WebsocketTutorialApiRole.
10. Selezionare Create Route (Crea route).

Successivamente, si crea un modello di mappatura per inviare i parametri della richiesta alla macchina a stati Step Functions.

Per creare un modello di mappatura

1. Scegli il percorso di invio del messaggio, quindi scegli la scheda Richiesta di integrazione.
2. Nella sezione Modelli di richiesta, scegli Modifica.
3. Per Espressione di selezione del modello, immettere**`\$default`**.
4. Scegli Modifica.
5. Nella sezione Richiedi modelli, scegli Crea modello.
6. Per la chiave modello, inserisci**`\$default`**.
7. Per Genera modello, inserisci il seguente modello di mappatura:

```
#set($domain = "$context.domainName")
#set($stage = "$context.stage")
#set($body = $input.json('$'))
#set($getMessage = $util.parseJson($body))
```

```
#set($mymessage = $getMessage.message)
{
"input": "{\"domain\": \"$domain\", \"stage\": \"$stage\", \"message\":
 \"$mymessage\"}",
"stateMachineArn": "arn:aws:states:us-east-2:123456789012:stateMachine:WebSocket-
Tutorial-StateMachine"
}
```

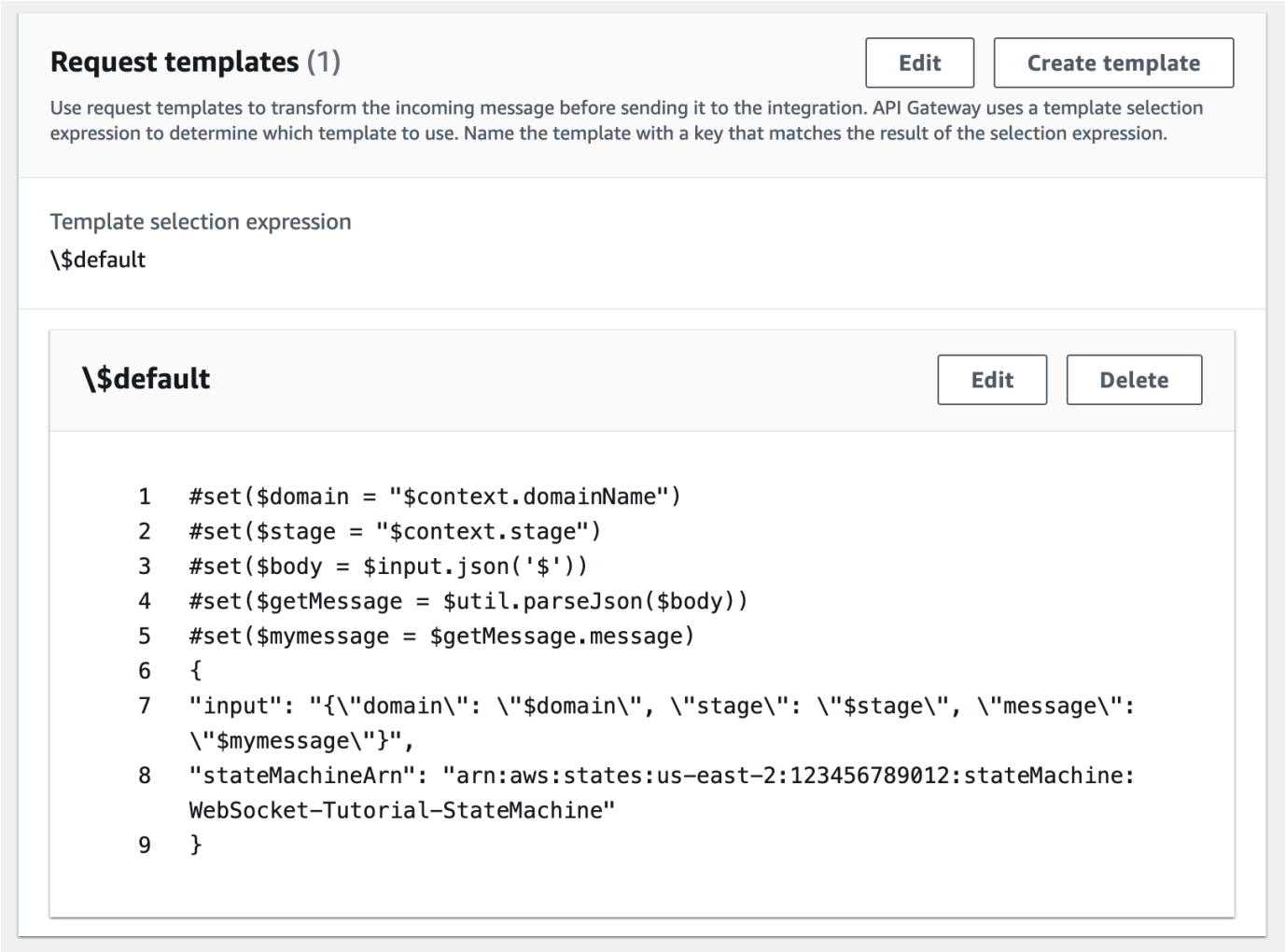
Sostituisci il *`stateMachineArn`* con ARN la macchina a stati creata da AWS CloudFormation.

Il modello di mappatura esegue le seguenti operazioni:

- Crea la variabile `$domain` utilizzando la variabile `domainName` di contesto.
- Crea la variabile `$stage` utilizzando la variabile di contesto`stage`.

  Le `$stage` variabili `$domain` and sono necessarie per creare un callbackURL.
- Riceve il `sendmessage` JSON messaggio in arrivo ed estrae la proprietà. `message`
- Crea l'input per la macchina a stati. L'input è il dominio e lo stadio del WebSocket API messaggio proveniente dal `sendmessage` percorso.

8. Scegli Crea modello.



È possibile creare un'integrazione non proxy sulle rotte $connect o $disconnect, per aggiungere o rimuovere direttamente un ID di connessione dalla tabella DynamoDB, senza richiamare una funzione Lambda.

Fase 6: Metti alla prova il tuo API

Successivamente, distribuirai e testerai il tuo API per assicurarti che funzioni correttamente. Utilizzerai il `wscat` comando per connetterti a API e poi, utilizzerai un comando barra per inviare un frame ping per verificare la connessione a. WebSocket API

Per distribuire il tuo API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel pannello di navigazione principale, scegli Percorsi.

3. Seleziona Deploy (Implementa)API.
4. Per Stage, scegli produzione.
5. (Facoltativo) Per la descrizione della distribuzione, inserire una descrizione.
6. Seleziona Deploy (Implementa).

Dopo aver distribuito il tuoAPI, puoi richiamarlo. Usa l'invoke URL per chiamare il tuo. API

Per ottenere l'invoce URL per il tuo API

1. Scegli il tuo API.
2. Scegli Stages (Fasi), quindi scegli production (produzione).
3. Nota i tuoiAPI. WebSocket URL URLDovrebbero assomigliare `awss://abcdef123.execute-api.us-east-2.amazonaws.com/production`.

Ora che hai la tua invokeURL, puoi testare la connessione al tuo WebSocket API.

Per testare la connessione al tuo API

1. Usa il seguente comando per connetterti al tuoAPI. Innanzitutto, verificate la connessione richiamando il `/ping` percorso.

```
wscat -c wss://abcdef123.execute-api.us-east-2.amazonaws.com/production -H
 "Authorization: Allow" --slash -P
```

```
Connected (press CTRL+C to quit)
```

2. Immettete il seguente comando per eseguire il ping del frame di controllo. È possibile utilizzare un frame di controllo per scopi keepalive dal lato client.

```
/ping
```

Il risultato sarà simile al seguente:

```
< Received pong (data: "")
```

Ora che hai testato la connessione, puoi verificare che API funzioni correttamente. In questo passaggio, si apre una nuova finestra di terminale in modo da WebSocket API poter inviare un messaggio a tutti i client connessi.

Per testare il tuo API

1. Apri un nuovo terminale ed esegui nuovamente il comando `wscat` con i parametri seguenti.

```
wscat -c wss://abcdef123.execute-api.us-east-2.amazonaws.com/production -H
 "Authorization: Allow"
```

```
Connected (press CTRL+C to quit)
```

2. APIGateway determina quale percorso richiamare in base all'espressione di selezione API della richiesta di percorso dell'utente. La tua espressione API di selezione del percorso è`$request.body.action`. Di conseguenza, API Gateway richiama il `sendmessage` percorso quando si invia il seguente messaggio:

```
{"action": "sendmessage", "message": "hello, from Step Functions!"}
```

   La macchina a stati Step Functions associata alla route richiama una funzione Lambda con il messaggio e il callback. URL La funzione Lambda richiama il API Gateway Management API e invia il messaggio a tutti i client connessi. Tutti i client ricevono il seguente messaggio:

```
< hello, from Step Functions!
```

Ora che hai testato il tuo WebSocket API, puoi disconnetterti dal tuoAPI.

Per disconnetterti dal tuo API

- Premi `CTRL+C` per disconnetterti dal tuo. API

  Quando un client si disconnette dal tuoAPI, API Gateway richiama la tua API route $disconnect. L'integrazione Lambda per la tua route $disconnect rimuove l'ID API di connessione da DynamoDB.

Fase 7: pulire

Per evitare costi non necessari, eliminare le risorse create nell'ambito di questo tutorial. I seguenti passaggi eliminano lo stack e. AWS CloudFormation WebSocket API

Per eliminare un WebSocket API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nella APIspagina, seleziona il tuo websocket-api.
3. Scegliere Azioni, scegliere Elimina, quindi confermare la scelta.

Per eliminare uno stack AWS CloudFormation

1. Apri la AWS CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformation.
2. Seleziona il tuo AWS CloudFormation stack.
3. Scegli Elimina e conferma la tua scelta.

Passaggi successivi

Puoi automatizzare la creazione e la pulizia di tutte le AWS risorse coinvolte in questo tutorial. Per un esempio di AWS CloudFormation modello che automatizza queste azioni per questo tutorial, vedi ws-sfn.zip.

API REST di API Gateway

Un'API REST in API Gateway è una raccolta di risorse e metodi integrati con endpoint HTTP di backend, funzioni Lambda o altri servizi. AWS Puoi utilizzare le caratteristiche API Gateway per semplificare tutti gli aspetti del ciclo di vita dell'API, dalla creazione al monitoraggio delle API di produzione.

Le API REST di API Gateway utilizzano un modello di richiesta/risposta in cui un client invia una richiesta a un servizio e il servizio risponde in modo sincrono. Questo tipo di modello è adatto per molti diversi tipi di applicazioni che dipendono dalla comunicazione sincrona.

Argomenti

- Sviluppa REST APIs in API Gateway
- Pubblica API REST che i clienti possono richiamare
- Ottimizzazione delle prestazioni delle API REST
- Distribuisci le tue API REST ai clienti in API Gateway
- Proteggi le tue API REST in API Gateway
- Monitora le API REST in API Gateway

Sviluppa REST APIs in API Gateway

In Amazon API Gateway, crei una REST API raccolta di entità programmabili note come risorse API Gateway. Ad esempio, usi una RestApirisorsa per rappresentare una API che può contenere una raccolta di entità Resource.

Ogni `Resource` entità può avere una o più risorse Method. A `Method` è una richiesta in entrata inviata dal client ed è espressa nei parametri e nel corpo della richiesta. Definisce l'interfaccia di programmazione dell'applicazione per consentire al client di accedere all'espostoResource. Per integrarlo `Method` con un endpoint di backend, noto anche come endpoint di integrazione, è necessario creare una risorsa di integrazione. Ciò inoltra la richiesta in arrivo a un endpoint di integrazione specificato. URI Se necessario, puoi trasformare i parametri della richiesta o il corpo della richiesta per soddisfare i requisiti del backend.

Per le risposte, è possibile creare una MethodResponserisorsa per rappresentare una risposta alla richiesta ricevuta dal client e creare una IntegrationResponserisorsa per rappresentare la risposta alla richiesta restituita dal backend. Puoi configurare la risposta di integrazione per trasformare i dati della

risposta di back-end prima di restituire i dati al client o passare la risposta di back-end al client senza modificarla.

Per aiutare i tuoi clienti a comprendere la tuaAPI, puoi anche fornire la documentazione relativaAPI, come parte della API creazione o dopo API la creazione. Per abilitare questa funzionalità, aggiungi una DocumentationPartrisorsa per un'APIentità supportata.

Per controllare il modo in cui i client chiamano unAPI, usa IAMle autorizzazioni, un autorizzatore Lambda o un pool di utenti Amazon Cognito. Per misurare l'utilizzo del tuoAPI, configura piani di utilizzo per limitare le richieste. API Puoi abilitarli durante la creazione o l'aggiornamento del tuo. API

Per un'introduzione su come creare unAPI, vederethe section called “Tutorial: Creare un'integrazione REST API con un proxy Lambda”. Per ulteriori informazioni sulle funzionalità di API Gateway che potresti utilizzare durante lo sviluppo di un RESTAPI, consulta i seguenti argomenti. Questi argomenti contengono informazioni e procedure concettuali che è possibile eseguire utilizzando la console API API Gateway RESTAPI, il Gateway AWS CLI, o uno dei AWS SDKs.

Argomenti

- Tipi di endpoint API per le API REST in API Gateway
- Metodi per REST APIs in API Gateway
- Controlla e gestisci l'accesso alle API REST in API Gateway
- Integrazioni per le API REST in API Gateway
- Richiedi la convalida per REST APIs in API Gateway
- Trasformazioni dei dati per REST APIs in Gateway API
- Risposte gateway per REST APIs in API Gateway
- CORSper REST APIs in API Gateway
- Tipi di supporti binari per REST APIs in API Gateway
- Richiama le API REST in API Gateway
- Sviluppa REST APIs utilizzando Open API in API Gateway

Tipi di endpoint API per le API REST in API Gateway

Per tipo di endpoint API si intende il nome host dell'API. Il tipo di endpoint di API può essere ottimizzato per i confini, regionale o privato, a seconda della provenienza della maggior parte del traffico dell'API.

Endpoint API ottimizzati per edge

Un endpoint API ottimizzato per l'edge in genere indirizza le richieste al CloudFront Point of Presence (POP) più vicino, il che può essere utile nei casi in cui i clienti sono distribuiti geograficamente. Questo è il tipo di endpoint di default per API REST API Gateway.

Le API ottimizzate per i confini utilizzano una lettera maiuscola iniziale per i nomi delle intestazioni HTTP, ad esempio `Cookie`.

CloudFront ordina i cookie HTTP in ordine naturale in base al nome del cookie prima di inoltrare la richiesta all'origine. Per ulteriori informazioni sul modo in cui CloudFront elabora i cookie, consulta Memorizzazione nella cache dei contenuti basati sui cookie.

Qualsiasi nome di dominio utilizzato per un'API con edge ottimizzato si applica in tutte le regioni.

Endpoint API regionali

Un endpoint API regionale è destinato ai clienti nella stessa regione. Quando un client in esecuzione su un'istanza EC2 chiama un'API nella stessa regione o quando un'API è destinata a servire un numero limitato di client con richieste elevate, un'API regionale riduce il sovraccarico di connessione.

Per un'API regionale, qualsiasi nome di dominio personalizzato utilizzato è specifico della regione in cui viene distribuita l'API. Se si distribuisce un'API regionale in più regioni, questa può avere lo stesso nome di dominio personalizzato in tutte le regioni. È possibile utilizzare domini personalizzati insieme ad Amazon Route 53 per eseguire attività come l'instradamento basato su latenza. Per ulteriori informazioni, consulta the section called “Configura un nome di dominio personalizzato regionale” e the section called “Configura un nome di dominio personalizzato ottimizzato per i dispositivi periferici in Gateway API”.

Gli endpoint di API regionali passano tutti i nomi di intestazione senza alcuna modifica.

> **Note**
>
> Nei casi in cui i client API siano distribuiti geograficamente, può comunque essere opportuno utilizzare un endpoint API regionale, insieme alla propria CloudFront distribuzione Amazon per garantire che API Gateway non associ l'API a distribuzioni controllate dal servizio. CloudFront Per ulteriori informazioni su questo caso d'uso, vedi Come posso configurare API Gateway con la mia CloudFront distribuzione? .

Endpoint API privati

Un endpoint di API privato è un endpoint di API al quale è possibile accedere solo da Amazon Virtual Private Cloud (VPC) utilizzando un endpoint VPC di interfaccia, ovvero un'interfaccia di rete dell'endpoint creato nel VPC. Per ulteriori informazioni, consulta the section called “Privato REST APIs”.

Gli endpoint API privati passano tutti i nomi delle intestazioni senza alcuna modifica.

Modifica di un tipo di endpoint API pubblico o privato in API Gateway

La modifica di un tipo di endpoint API richiede l'aggiornamento della configurazione dell'API. Puoi modificare un tipo di API esistente utilizzando la console API Gateway AWS CLI, o un AWS SDK per API Gateway. Il tipo di endpoint non potrà essere modificato fino a quando non viene completata la modifica corrente ma, durante tale periodo, l’API sarà disponibile.

Sono supportate le seguenti modifiche ai tipi di endpoint:

- Da ottimizzato per i dispositivi perimetrali a regionali o privati
- Da regionale a ottimizzato per l'edge o privato
- Da privato a regionale

Non è possibile modificare un'API privata in un'API ottimizzata per i confini.

Se stai modificando un'API pubblica da ottimizzata per i bordi a regionale o viceversa, tieni presente che un'API ottimizzata per i bordi può avere comportamenti diversi rispetto a un'API regionale. Ad esempio, un'API ottimizzata per i confini rimuove l'intestazione `Content-MD5`. Qualsiasi valore hash MD5 passato al back-end può essere espresso in un parametro stringa della richiesta o in una proprietà del corpo. Tuttavia, l'API regionale trasmette questa intestazione, sebbene possa rimappare il nome dell'intestazione con un altro nome. Comprendere le differenze aiuta a decidere come aggiornare un'API ottimizzata per i bordi a una regionale o da un'API regionale a un'API ottimizzata per i bordi.

Argomenti

- Uso della console API Gateway per modificare un tipo di endpoint API
- Utilizza il per modificare il tipo AWS CLI di endpoint dell'API

Uso della console API Gateway per modificare un tipo di endpoint API

Per modificare il tipo di endpoint API della tua API, esegui uno dei seguenti insiemi di passaggi:

Conversione di un endpoint pubblico da regionale o ottimizzato per l'edge e viceversa

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere una REST API.
3. Scegli Impostazioni API.
4. Nella sezione Dettagli API, scegli Modifica.
5. In Tipo di endpoint API, seleziona Ottimizzato per l'edge o Regionale.
6. Seleziona Salvataggio delle modifiche.
7. Ridistribuisci la tua API in modo che le modifiche diventino effettive.

Per convertire un endpoint privato in un endpoint regionale

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere un'API REST.
3. Modifica la policy delle risorse per la tua API per rimuovere qualsiasi menzione di VPC o endpoint VPC in modo che le chiamate alle API all'esterno del VPC, nonché all'interno del VPC, verranno eseguite correttamente.
4. Scegli Impostazioni API.
5. Nella sezione Dettagli API, scegli Modifica.
6. In Tipo di endpoint, scegliRegionale.
7. Seleziona Salvataggio delle modifiche.
8. Rimuovi la policy delle risorse dall'API.
9. Ridistribuisci la tua API in modo che le modifiche diventino effettive.

Per convertire un endpoint regionale in un endpoint privato

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere una REST API.
3. Crea una politica delle risorse che garantisca l'accesso al tuo VPC o endpoint VPC. Per ulteriori informazioni, consulta ???.

4. Scegli Impostazioni API.
5. Nella sezione Dettagli API, scegli Modifica.
6. Per Tipo di endpoint API scegli Privato.
7. (Facoltativo) Per gli ID degli endpoint VPC, seleziona gli ID degli endpoint VPC che desideri associare alla tua API privata.
8. Seleziona Salvataggio delle modifiche.
9. Ridistribuisci la tua API in modo che le modifiche diventino effettive.

Utilizza il per modificare il tipo AWS CLI di endpoint dell'API

Per utilizzare il per aggiornare un'API ottimizzata AWS CLI per i dispositivi periferici il cui ID API è, chiama quanto segue: *{api-id}* update-rest-api

```
aws apigateway update-rest-api \
    --rest-api-id {api-id} \
    --patch-operations op=replace,path=/endpointConfiguration/types/EDGE,value=REGIONAL
```

La risposta di esito positivo ha il codice di stato `200 OK` e un payload simile al seguente:

```
{

    "createdDate": "2017-10-16T04:09:31Z",
    "description": "Your first API with Amazon API Gateway. This is a sample API that
 integrates via HTTP with our demo Pet Store endpoints",
    "endpointConfiguration": {
        "types": "REGIONAL"
    },
    "id": "0gsnjtjck8",
    "name": "PetStore imported as edge-optimized"
}
```

Al contrario, aggiorna un'API regionale in API ottimizzata per i confini in questo modo:

```
aws apigateway update-rest-api \
    --rest-api-id {api-id} \
    --patch-operations op=replace,path=/endpointConfiguration/types/REGIONAL,value=EDGE
```

Poiché put-rest-apiserve per aggiornare le definizioni delle API, non è applicabile all'aggiornamento di un tipo di endpoint API.

Metodi per REST APIs in API Gateway

In API Gateway, un API metodo incorpora una richiesta di metodo e una risposta al metodo. Si imposta un API metodo per definire cosa deve o deve fare un client per inviare una richiesta di accesso al servizio dal backend e per definire le risposte che il client riceve in cambio. Per l'input puoi scegliere i parametri di richiesta del metodo o un payload applicabile per consentire al client di fornire i dati obbligatori o facoltativi al runtime. Per l'output si specifica il codice di stato della risposta del metodo, le intestazioni e il corpo applicabile come destinazioni a cui mappare i dati delle risposte di back-end prima che vengano restituite al client. Per aiutare lo sviluppatore del client a comprendere i comportamenti e i formati di input e output del clientAPI, è possibile documentare API e fornire messaggi di errore corretti per le richieste non valide.

Una richiesta API di metodo è una HTTP richiesta. Per impostare la richiesta del metodo, configurate un HTTP metodo (o verbo), il percorso di una API risorsa, le intestazioni, i parametri della stringa di query applicabili. È inoltre possibile configurare un payload quando il HTTP metodo è`POST`, `PUT` o. `PATCH` Ad esempio, per recuperare un animale domestico utilizzando il PetStore campione API, è necessario definire il API metodo request of`GET /pets/{petId}`, dove `{petId}` è un parametro di percorso che può richiedere un numero in fase di esecuzione.

```
GET /pets/1
Host: apigateway.us-east-1.amazonaws.com
...
```

Se il client specifica un percorso non corretto, ad esempio `/pet/1` o `/pets/one` anziché `/pets/1`, viene generata un'eccezione.

Una risposta al API metodo è una HTTP risposta con un determinato codice di stato. Per un'integrazione non proxy, è necessario configurare le risposte di metodo per specificare le destinazioni obbligatorie o facoltative delle mappature. Queste trasformano il corpo o le intestazioni delle risposte di integrazione nel corpo o nelle intestazioni delle risposte del metodo associato. La mappatura può essere semplice come la trasformazione di un'identità che trasferisce le intestazioni o il corpo tramite l'integrazione senza alcuna modifica. Ad esempio, la risposta del metodo `200` seguente mostra l'esempio di una risposta di integrazione riuscita che viene passata senza essere modificata.

```
200 OK
Content-Type: application/json
...
```

```
{
    "id": "1",
    "type": "dog",
    "price": "$249.99"
}
```

Inizialmente puoi definire una risposta di metodo che corrisponde a una risposta specifica dal back-end. Generalmente vengono utilizzate le risposte 2XX, 4XX e 5XX. Tuttavia questo approccio potrebbe essere poco pratico perché non sempre si conoscono in anticipo le risposte che un back-end potrebbe restituire. In pratica, è possibile designare una sola risposta del metodo come predefinita per gestire le risposte non note o non mappate dal back-end. È buona norma designare la risposta 500 come predefinita. In ogni caso, devi configurare almeno una risposta del metodo per le integrazioni non proxy. In caso contrario, API Gateway restituisce una risposta di errore 500 al client anche quando la richiesta ha esito positivo nel backend.

Per supportare un linguaggio fortemente tipizzatoSDK, ad esempio Java SDKAPI, è necessario definire il modello di dati per l'input per la richiesta del metodo e definire il modello di dati per l'output della risposta del metodo.

Prerequisiti

Prima di configurare un API metodo, verificate quanto segue:

- È necessario che il metodo sia disponibile in API Gateway. Segui le istruzioni in Tutorial: creare un file REST API con un'HTTPintegrazione non proxy.
- Se desideri che il metodo comunichi con una funzione Lambda, devi aver già creato il ruolo di invocazione Lambda e il ruolo di esecuzione Lambda in. IAM Devi aver creato anche la funzione Lambda con cui il metodo comunicherà in AWS Lambda. Per creare ruoli e funzioni, usa le istruzioni disponibili nella sezione Creazione di una funzione Lambda per l'integrazione non proxy Lambda dell'argomento Scegli un tutorial di integrazione AWS Lambda.
- Se desideri che il metodo comunichi con un'integrazione HTTP o HTTP proxy, devi aver già creato e avere accesso all'HTTPendpoint URL con cui il metodo comunicherà.
- Verifica che i certificati HTTP e gli endpoint HTTP proxy siano supportati da API Gateway. Per dettagli, consulta APIAutorità di certificazione supportate dal gateway HTTP e integrazioni HTTP proxy in Gateway API.

> **Note**
>
> Quando crei un metodo utilizzando la REST API console, configuri sia la richiesta di integrazione che la richiesta del metodo. Per ulteriori informazioni, consulta the section called " Configurazione di una richiesta di integrazione tramite la console".

Argomenti

- Imposta una richiesta di metodo in API Gateway
- Imposta un metodo di risposta in Gateway API
- Configurare un metodo utilizzando la console API Gateway

Imposta una richiesta di metodo in API Gateway

La configurazione di una richiesta di metodo implica l'esecuzione delle seguenti attività, dopo aver creato una RestApirisorsa:

1. Creare una nuova entità Resource API o scegliere un'entità API Resource esistente.
2. Creazione di una risorsa API Method che sia un HTTP verbo specifico sulla nuova o sulla scelta APIResource. Puoi suddividere ulteriormente questa attività svolgendo le operazioni secondarie seguenti:
   - Aggiungere un HTTP metodo alla richiesta del metodo
   - Configura i parametri di richiesta
   - Definisci un modello per il corpo della richiesta
   - Attua uno schema di autorizzazione
   - Abilita la convalida delle richieste

Puoi eseguire queste attività utilizzando i metodi seguenti:

- APIConsole gateway
- AWS CLI comandi (create-resource e put-method)
- AWS SDKfunzioni (ad esempio, in Node.js e) createResourceputMethod
- APIGateway REST API (resource:create e method:put).

Argomenti

- APIConfigura le risorse
- Imposta un HTTP metodo
- Configurazione dei parametri di richiesta del metodo
- Imposta un modello di richiesta del metodo
- Configurazione dell'autorizzazione della richiesta del metodo
- Configurazione della convalida della richiesta del metodo

APIConfigura le risorse

In un API GatewayAPI, esponete le risorse indirizzabili come un albero di entità API Resources, con la root resource (`/`) in cima alla gerarchia. La risorsa root è relativa alla API baseURL, che consiste nell'APIendpoint e nel nome dello stadio. Nella console API Gateway, questa base URI è denominata Invoke URI e viene visualizzata nell'editor API dello stage dopo la distribuzioneAPI.

L'APIendpoint può essere un nome host predefinito o un nome di dominio personalizzato. Il formato del nome host predefinito è il seguente:

```
{api-id}.execute-api.{region}.amazonaws.com
```

In questo formato, *`{api-id}`* rappresenta l'APIidentificatore generato da API Gateway. La *`{region}`* variabile rappresenta la AWS regione (ad esempio`us-east-1`) scelta durante la API creazione di. Un nome di dominio personalizzato è un nome intuitivo in un dominio Internet valido. Ad esempio se hai registrato un dominio Internet di `example.com`, qualsiasi `*.example.com` è un nome di dominio personalizzato valido. Per ulteriori informazioni, consulta l'argomento relativo alla creazione di un nome di dominio personalizzato.

Per l'PetStore esempio API, la risorsa root (`/`) espone il negozio di animali domestici. La risorsa `/pets` rappresenta l'insieme di animali domestici disponibili nel negozio. `/pets/{petId}` espone un singolo animale domestico di un identificatore specificato (`petId`). Il parametro di percorso di `{petId}` fa parte dei parametri di richiesta.

Per configurare una API risorsa, scegliete una risorsa esistente come risorsa principale e quindi create la risorsa figlia sotto questa risorsa principale. Inizia con la risorsa radice come padre, aggiungi una risorsa alla risorsa padre, aggiungi un'alta risorsa a questa risorsa figlio come nuovo padre e così via fino all'identificatore del padre. Quindi aggiungi la risorsa con nome alla risorsa padre.

Con AWS CLI, puoi chiamare il `get-resources` comando per scoprire quali risorse di una API sono disponibili:

```
aws apigateway get-resources --rest-api-id <apiId> \
                             --region <region>
```

Il risultato è un elenco delle risorse attualmente disponibili diAPI. Per il nostro PetStore esempioAPI, questo elenco è simile al seguente:

```
{
    "items": [
        {
            "path": "/pets",
            "resourceMethods": {
                "GET": {}
            },
            "id": "6sxz2j",
            "pathPart": "pets",
            "parentId": "svzr2028x8"
        },
        {
            "path": "/pets/{petId}",
            "resourceMethods": {
                "GET": {}
            },
            "id": "rjkmth",
            "pathPart": "{petId}",
            "parentId": "6sxz2j"
        },
        {
            "path": "/",
            "id": "svzr2028x8"
        }
    ]
}
```

Ogni voce elenca gli identificatori della risorsa (`id`) e, ad eccezione della risorsa radice, la rispettiva risorsa padre più prossima (`parentId`), nonché il nome della risorsa (`pathPart`). La risorsa radice ha la particolarità di non avere una risorsa padre. Dopo avere scelto una risorsa come padre, chiama il comando seguente per aggiungere una risorsa figlio.

```
aws apigateway create-resource --rest-api-id <apiId> \
```

```
                                --region <region> \
                                --parent-id <parentId> \
                                --path-part <resourceName>
```

Ad esempio, per aggiungere cibo per animali domestici in vendita sul PetStore sito Web, aggiungi una `food` risorsa alla radice (/) `path-part` impostando su `food` e `parent-id` `susvzr2028x8`. Il risultato avrà il seguente aspetto:

```
{
    "path": "/food",
    "pathPart": "food",
    "id": "xdsvhp",
    "parentId": "svzr2028x8"
}
```

Utilizzate una risorsa proxy per API semplificare la configurazione

Man mano che l'attività cresce, il PetStore proprietario può decidere di mettere in vendita cibo, giocattoli e altri articoli relativi agli animali domestici. A questo scopo, puoi aggiungere `/food`, `/toys` e altre risorse sotto la risorsa radice. Puoi anche aggiungere altre risorse sotto ogni categoria di vendita, ad esempio `/food/{type}/{item}`, `/toys/{type}/{item}` e così via. Questa operazione può risultare tediosa. Se si decide di aggiungere un livello intermedio `{subtype}` ai percorsi delle risorse per modificare la gerarchia dei percorsi`/food/{type}/{subtype}/{item}`, ecc.`/toys/{type}/{subtype}/{item}`, le modifiche interromperanno la configurazione esistenteAPI. Per evitare ciò, è possibile utilizzare una risorsa proxy API Gateway per esporre un set di API risorse tutte contemporaneamente.

APIGateway definisce una risorsa proxy come segnaposto per una risorsa da specificare al momento dell'invio della richiesta. È possibile esprimere una risorsa proxy mediante uno speciale parametro di percorso `{proxy+}`, spesso definito parametro di percorso greedy. Il segno + indica le risorse figlio aggiunte. Il segnaposto `/parent/{proxy+}` indica qualsiasi risorsa che corrisponda al modello di percorso di `/parent/*`. Il nome del parametro di percorso greedy, `proxy`, può essere sostituito da un'altra stringa, come qualsiasi nome di parametro di percorso.

Utilizzando il AWS CLI, si chiama il seguente comando per configurare una risorsa proxy sotto root ()`/{proxy+}`:

```
aws apigateway create-resource --rest-api-id <apiId> \
                                --region <region> \
```

```
                                --parent-id <rootResourceId> \
                                --path-part {proxy+}
```

Il risultato è simile al seguente:

```
{
    "path": "/{proxy+}",
    "pathPart": "{proxy+}",
    "id": "234jdr",
    "parentId": "svzr2028x8"
}
```

`PetStoreAPI`Ad esempio, è possibile utilizzare `/{proxy+}` per rappresentare `/pets` sia la che`/pets/{petId}`. Questa risorsa proxy può anche fare riferimento a qualsiasi altra risorsa (esistente o to-be-added) `/food/{type}/{item}/toys/{type}/{item}`, ad esempio, ecc.`/food/{type}/{subtype}/{item}`, oppure`/toys/{type}/{subtype}/{item}`, ecc. Lo sviluppatore del back-end stabilisce la gerarchia delle risorse, mentre allo sviluppatore del client spetta il compito di comprenderla. APIGateway trasmette semplicemente tutto ciò che il client ha inviato al backend.

Un API può avere più di una risorsa proxy. Ad esempio, le seguenti risorse proxy sono consentite all'interno di unaAPI, supponendo che non `/parent/{proxy+}` sia la stessa fonte principale di. `/parent/{child}/{proxy+}`

```
/{proxy+}
/parent/{proxy+}
/parent/{child}/{proxy+}
```

Quando una risorsa proxy presenta elementi di pari livello non proxy, le risorse di tali elementi vengono escluse dalla rappresentazione della risorsa proxy. Per gli esempi precedenti, `/{proxy+}` si riferisce a qualsiasi risorsa al di sotto della risorsa radice, ad eccezione delle risorse `/parent[/*]`. In altri termini, una richiesta di metodo a una risorsa specifica ha la precedenza rispetto a una richiesta di metodo a una risorsa generica allo stesso livello della gerarchia di risorse.

Una risorsa proxy non può avere risorse figlio. Qualsiasi API risorsa successiva `{proxy+}` è ridondante e ambigua. Le seguenti risorse proxy non sono consentite all'interno di un. API

```
/{proxy+}/child
/parent/{proxy+}/{child}
/parent/{child}/{proxy+}/{grandchild+}
```

Imposta un HTTP metodo

Una richiesta di API metodo è incapsulata dalla risorsa API Gateway Method. Per impostare la richiesta del metodo, è necessario innanzitutto creare un'istanza della `Method` risorsa, impostando almeno un HTTP metodo e un tipo di autorizzazione sul metodo.

Strettamente associato alla risorsa proxy, API Gateway supporta un HTTP metodo di. `ANY` Questo `ANY` metodo rappresenta qualsiasi HTTP metodo che deve essere fornito in fase di esecuzione. Consente di utilizzare un'unica configurazione di API metodo per tutti i HTTP metodi supportati di`DELETE`,`GET`,`HEAD`, `OPTIONSPATCH`,`POST`, e`PUT`.

Puoi configurare il metodo `ANY` anche per una risorsa non proxy. Combinando il `ANY` metodo con una risorsa proxy, si ottiene una configurazione a API metodo singolo per tutti i HTTP metodi supportati rispetto a tutte le risorse di unAPI. Inoltre, il backend può evolversi senza interrompere la configurazione esistenteAPI.

Prima di configurare un API metodo, considera chi può chiamarlo. Imposta il tipo di autorizzazione in base al tuo piano. Per l'accesso aperto, impostalo su `NONE`. Per utilizzare IAM le autorizzazioni, imposta il tipo di autorizzazione su`AWS_IAM`. Per usare una funzione di autorizzazione Lambda, imposta questa proprietà su `CUSTOM`. Per utilizzare un pool di utenti di Amazon Cognito, imposta il tipo di autorizzazione su `COGNITO_USER_POOLS`.

Il AWS CLI comando seguente mostra come creare una richiesta di metodo del `ANY` verbo su una risorsa specificata (`6sxz2j`), utilizzando IAM le autorizzazioni per controllarne l'accesso.

```
aws apigateway put-method --rest-api-id vaz7da96z6 \
       --resource-id 6sxz2j \
       --http-method ANY \
       --authorization-type AWS_IAM \
       --region us-west-2
```

Per creare una richiesta di API metodo con un tipo di autorizzazione diverso, vedere. the section called “Configurazione dell'autorizzazione della richiesta del metodo”

Configurazione dei parametri di richiesta del metodo

I parametri di richiesta del metodo consentono a un client di fornire i dati di input o il contesto di esecuzione necessari per completare la richiesta del metodo. Un parametro di metodo può essere un parametro di percorso, un'intestazione o un parametro stringa di query. Nell'ambito della configurazione della richiesta del metodo, è necessario dichiarare i parametri di richiesta obbligatori

per renderli disponibili per il client. Per l'integrazione non proxy, puoi convertire questi parametri di richiesta in un formato compatibile con i requisiti di back-end.

Ad esempio, per la richiesta del metodo `GET /pets/{petId}`, la variabile di percorso `{petId}` è un parametro di richiesta obbligatorio. Puoi dichiarare questo parametro di percorso quando chiami il comando `put-method` di AWS CLI, come mostrato di seguito:

```
aws apigateway put-method --rest-api-id vaz7da96z6 \
       --resource-id rjkmth \
       --http-method GET \
       --authorization-type "NONE" \
       --region us-west-2 \
       --request-parameters method.request.path.petId=true
```

Se un parametro non è obbligatorio, puoi impostarlo su `false` in `request-parameters`. Ad esempio, se il `GET /pets` metodo utilizza un parametro di stringa di query opzionale di e un parametro di `type` intestazione opzionale di`breed`, è possibile dichiararli utilizzando il CLI comando seguente, presupponendo che la `/pets` risorsa sia: `id 6sxz2j`

```
aws apigateway put-method --rest-api-id vaz7da96z6 \
       --resource-id 6sxz2j \
       --http-method GET \
       --authorization-type "NONE" \
       --region us-west-2 \
       --request-parameters
 method.request.querystring.type=false,method.request.header.breed=false
```

Invece di questa forma abbreviata, è possibile utilizzare una JSON stringa per impostare il valore: `request-parameters`

```
'{"method.request.querystring.type":false,"method.request.header.breed":false}'
```

Con questa configurazione, il client può eseguire query sugli animali domestici in base al tipo:

```
GET /pets?type=dog
```

Il client può inoltre eseguire query sui cani di razza barboncino come segue:

```
GET /pets?type=dog
```

```
breed:poodle
```

Per informazioni su come mappare i parametri di richiesta dei metodi ai parametri di richiesta di integrazione, consulta the section called “Integrazioni”.

Imposta un modello di richiesta del metodo

Per un API metodo in grado di accettare dati di input in un payload, puoi utilizzare un modello. Un modello è espresso in una bozza JSON dello schema 4 e descrive la struttura dei dati del corpo della richiesta. Utilizzando un modello, un client può stabilire come costruire un payload di richiesta del metodo come input. Ancora più importante, API Gateway utilizza il modello per convalidare una richiesta, generare un SDK modello di mappatura e inizializzare un modello di mappatura per configurare l'integrazione nella console Gateway. API Per informazioni su come creare un modello, consultare Informazioni sui modelli di dati.

A seconda del tipo di contenuto, un payload del metodo può avere formati diversi. Un modello viene indicizzato in base al tipo di supporto del payload applicato. APIGateway utilizza l'intestazione della `Content-Type` richiesta per determinare il tipo di contenuto. Per impostare i modelli di richiesta del metodo, aggiungi coppie chiave-valore del `"<media-type>":"<model-name>"` formato alla `requestModels` mappa quando chiami il comando. AWS CLI `put-method`

Per utilizzare lo stesso modello indipendentemente dal tipo di contenuto, specifica `$default` come chiave.

Ad esempio, per impostare un modello sul JSON payload della richiesta del `POST /pets` metodo dell' PetStore esempioAPI, puoi chiamare il seguente comando: AWS CLI

```
aws apigateway put-method \
       --rest-api-id vaz7da96z6 \
       --resource-id 6sxz2j \
       --http-method POST \
       --authorization-type "NONE" \
       --region us-west-2 \
       --request-models '{"application/json":"petModel"}'
```

In questo esempio `petModel` è il valore della proprietà `name` di una risorsa `Model` che descrive un animale domestico. L'effettiva definizione dello schema è espressa come valore di JSON stringa della `schema`proprietà della `Model` risorsa.

In un Java, o in un altro formato fortemente tipizzato SDKAPI, i dati di input vengono espressi come `petModel` classe derivata dalla definizione dello schema. Con il modello di richiesta, i dati di input

generati SDK vengono inseriti nella `Empty` classe, che deriva dal `Empty` modello predefinito. In questo caso il client non può creare istanze della classe di dati corretta per fornire l'input richiesto.

Configurazione dell'autorizzazione della richiesta del metodo

Per controllare chi può chiamare il API metodo, puoi configurare il tipo di autorizzazione sul metodo. Puoi usare questo tipo per attivare uno degli autorizzatori supportati, inclusi IAM ruoli e politiche (`AWS_IAM`), un pool di utenti Amazon Cognito `COGNITO_USER_POOLS` () o un autorizzatore Lambda (). `CUSTOM`

Per utilizzare IAM le autorizzazioni per autorizzare l'accesso al API metodo, imposta la proprietà input su. `authorization-type` **`AWS_IAM`** Quando imposti questa opzione, API Gateway verifica la firma del chiamante sulla richiesta in base alle credenziali del chiamante. Se l'utente verificato dispone dell'autorizzazione per chiamare il metodo, la richiesta viene accettata. In caso contrario, la richiesta viene rifiutata e il chiamante riceve una risposta di errore. La chiamata al metodo non ha esito positivo a meno che il chiamante non sia autorizzato a richiamare il metodo. API La seguente IAM politica concede l'autorizzazione al chiamante di chiamare qualsiasi API metodo creato all'interno dello stesso: Account AWS

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "execute-api:Invoke"
            ],
            "Resource": "arn:aws:execute-api:*:*:*"
        }
    ]
}
```

Per ulteriori informazioni, consulta the section called “Usa le IAM autorizzazioni”.

Attualmente, puoi concedere questa politica solo agli utenti, ai gruppi e ai ruoli all'interno del API proprietario. Account AWS Gli utenti di un altro Account AWS soggetto possono chiamare i API metodi solo se sono autorizzati ad assumere un ruolo all'interno del API proprietario Account AWS con le autorizzazioni necessarie per avviare l'`execute-api:Invoke`azione. Per informazioni sulle autorizzazioni per più account, consulta Utilizzo dei ruoli. IAM

Puoi utilizzare AWS CLI un o un REST API client AWS SDK, come Postman, che implementa la firma Signature Version 4 (SigV4).

Per utilizzare un autorizzatore Lambda per autorizzare l'accesso al API metodo, imposta la proprietà `authorization-type` input su `CUSTOM` e imposta la proprietà `authorizer-id`input sul valore della `id`proprietà di un autorizzatore Lambda già esistente. L'autorizzazione Lambda di riferimento può essere di tipo `TOKEN` o `REQUEST`. Per informazioni su come creare un'autorizzazione Lambda, consulta the section called “Uso di autorizzazioni Lambda”.

Per utilizzare un pool di utenti di Amazon Cognito per autorizzare l'accesso al API metodo, imposta la proprietà `authorization-type` input su `COGNITO_USER_POOLS` e imposta la proprietà `authorizer-id`input sul valore della `id`proprietà dell'`COGNITO_USER_POOLS`autorizzatore già creato. Per informazioni sulla creazione di un'autorizzazione per il bacino d’utenza di Amazon Cognito, consulta the section called “Utilizza un pool di utenti di Amazon Cognito come autorizzazione per un'API REST”.

Configurazione della convalida della richiesta del metodo

Puoi abilitare la convalida della richiesta quando configuri una richiesta di metodo. API È prima necessario creare un validatore di richieste:

```
aws apigateway create-request-validator \
    --rest-api-id 7zw9uyk9kl \
    --name bodyOnlyValidator \
    --validate-request-body  \
    --no-validate-request-parameters
```

Questo CLI comando crea un validatore di richieste solo per il corpo. Di seguito è riportato l'output di esempio:

```
{
    "validateRequestParameters": false,
    "validateRequestBody": true,
    "id": "jgpyy6",
    "name": "bodyOnlyValidator"
}
```

Con questo validatore puoi abilitare la convalida delle richieste nell'ambito della configurazione della richiesta del metodo:

```
aws apigateway put-method \
    --rest-api-id 7zw9uyk9kl
    --region us-west-2
    --resource-id xdsvhp
    --http-method PUT
    --authorization-type "NONE"
    --request-parameters '{"method.request.querystring.type": false,
 "method.request.querystring.page":false}'
    --request-models '{"application/json":"petModel"}'
    --request-validator-id jgpyy6
```

Per essere incluso in una convalida, un parametro di richiesta deve essere dichiarato come obbligatorio. Se il parametro della stringa di query per la pagina viene usato nella convalida della richiesta, la mappa `request-parameters` dell'esempio precedente deve essere specificata come `'{"method.request.querystring.type": false, "method.request.querystring.page":true}'`.

Imposta un metodo di risposta in Gateway API

Una risposta al API metodo incapsula l'output di una richiesta di API metodo che il client riceverà. I dati di output includono un codice di HTTP stato, alcune intestazioni e possibilmente un corpo.

Con le integrazioni non proxy, i parametri di risposta specificati e il corpo possono essere mappati dai dati della risposta di integrazione associati oppure possono essere assegnati determinati valori statici in base alle mappature. Queste mappature vengono specificate nella risposta di integrazione. La mappatura può essere una trasformazione identica che passa attraverso l'integrazione senza alcuna modifica.

Con un'integrazione proxy, API Gateway trasmette automaticamente la risposta del backend alla risposta del metodo. Non è necessario impostare il API metodo di risposta. Tuttavia, con l'integrazione del proxy Lambda, la funzione Lambda deve restituire un risultato di questo formato di output affinché API Gateway mappi correttamente la risposta di integrazione a una risposta del metodo.

A livello di programmazione, la configurazione della risposta del metodo equivale alla creazione di una MethodResponserisorsa di API Gateway e all'impostazione delle proprietà di statusCode, e. responseParametersresponseModels

Quando si impostano i codici di stato per un API metodo, è necessario sceglierne uno come predefinito per gestire qualsiasi risposta di integrazione di un codice di stato imprevisto. È accettabile

impostare `500` come valore predefinito perché equivale a trasmettere risposte non mappate come errore del server. Per motivi didattici, la console API Gateway imposta la `200` risposta come predefinita. Tuttavia puoi reimpostare il valore su `500`.

Per configurare una risposta di metodo, devi aver creato la richiesta.

Configurazione dello stato del codice di una risposta di metodo

Lo stato del codice di una risposta di metodo definisce un tipo di risposta. Ad esempio, le risposte 200, 400 e 500 indicano rispettivamente riposte completate, errore lato client ed errore lato server.

Per impostare un codice di stato della risposta del metodo, imposta la `statusCode`proprietà su un codice di HTTP stato. Il comando AWS CLI seguente crea la risposta del metodo `200`.

```
aws apigateway put-method-response \
       --region us-west-2  \
       --rest-api-id vaz7da96z6 \
       --resource-id 6sxz2j \
       --http-method GET \
       --status-code 200
```

Configurazione dei parametri di risposta del metodo

I parametri di risposta del metodo definiscono le intestazioni che il client deve ricevere in risposta alla richiesta di metodo associata. I parametri di risposta specificano anche una destinazione a cui API Gateway mappa un parametro di risposta di integrazione, in base alle mappature prescritte nella risposta di integrazione del API metodo.

Per configurare i parametri di risposta del metodo, aggiungi alla mappa `responseParameters` di `MethodResponse` coppie chiave-valore nel formato `"{parameter-name}":"{boolean}"`. Il CLI comando seguente mostra un esempio di impostazione dell'`my-header`intestazione.

```
aws apigateway put-method-response \
        --region us-west-2 \
        --rest-api-id vaz7da96z6 \
        --resource-id 6sxz2j \
        --http-method GET \
        --status-code 200  \
        --response-parameters method.response.header.my-header=false
```

Configurazione dei modelli di risposta del metodo

Un modello di risposta del metodo definisce un formato del corpo della risposta. Prima di configurare il modello di risposta, è necessario creare il modello in API Gateway. A questo scopo, chiama il comando `create-model`. L'esempio seguente mostra come creare un modello `PetStorePet` per descrivere il corpo della risposta nella richiesta del metodo `GET /pets/{petId}`.

```
aws apigateway create-model \
    --region us-west-2 \
    --rest-api-id vaz7da96z6 \
    --content-type application/json \
    --name PetStorePet \
    --schema '{ \
                  "$schema": "http://json-schema.org/draft-04/schema#", \
                  "title": "PetStorePet", \
                  "type": "object", \
                  "properties": { \
                    "id": { "type": "number" }, \
                    "type": { "type": "string" }, \
                    "price": { "type": "number" } \
                  } \
                }'
```

Il risultato viene creato come `Model`risorsa API Gateway.

Per configurare i modelli di risposta del metodo per definire il formato del payload, aggiungi la coppia chiave-valore «application/json»:» PetStorePet "alla mappa della risorsa. `requestModelsMethodResponse` Il seguente AWS CLI comando mostra come eseguire questa operazione`put-method-response`:

```
aws apigateway put-method-response \
        --region us-west-2 \
        --rest-api-id vaz7da96z6 \
        --resource-id 6sxz2j \
        --http-method GET \
        --status-code 200  \
        --response-parameters method.response.header.my-header=false \
        --response-models '{"application/json":"PetStorePet"}'
```

La configurazione di un modello di risposta del metodo è necessaria quando si genera un codice fortemente digitato SDK per. API Garantisce che l'output venga trasmesso in una classe appropriata in Java o Objective-C. In altri casi l'impostazione di un modello è facoltativa.

Configurare un metodo utilizzando la console API Gateway

Quando si crea un metodo utilizzando la REST API console, si configurano sia la richiesta di integrazione che la richiesta del metodo. Per impostazione predefinita, API Gateway crea il `200` metodo di risposta per il metodo.

Le seguenti istruzioni mostrano come modificare le impostazioni di richiesta del metodo e come creare risposte aggiuntive per il metodo.

Argomenti

- Modifica una richiesta del metodo API API Gateway nella console Gateway
- Imposta una risposta del metodo API Gateway utilizzando la console Gateway API

Modifica una richiesta del metodo API API Gateway nella console Gateway

Queste istruzioni presuppongono che tu abbia già creato la tua richiesta di metodo. Per ulteriori informazioni su come creare un metodo, vederethe section called “ Configurazione di una richiesta di integrazione tramite la console”.

1. Nel riquadro Risorse, scegliete il metodo, quindi scegliete la scheda Richiesta metodo.
2. Nella sezione Impostazioni della richiesta del metodo scegli Modifica.
3. Per Autorizzazione seleziona un sistema di autorizzazione disponibile.

   a. Per abilitare l'accesso aperto al metodo per qualsiasi utente, scegli Nessuno. Questa fase può essere ignorata se l'impostazione predefinita non è stata modificata.

   b. Per utilizzare IAM le autorizzazioni per controllare l'accesso del client al metodo, selezionate`AWS_IAM`. Con questa scelta, solo gli utenti dei IAM ruoli a cui è associata la IAM politica corretta possono chiamare questo metodo.

      Per creare il IAM ruolo, specifica una politica di accesso con un formato simile al seguente:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
```

```
      "Resource": [
        "resource-statement"
      ]
    }
  ]
}
```

In questa politica di accesso, `resource-statement` è ARN il tuo metodo. È possibile trovare il ARN metodo desiderato selezionando il metodo nella pagina Risorse. Per ulteriori informazioni sull'impostazione delle IAM autorizzazioni, consultaControlla l'accesso a un REST API con IAM autorizzazioni.

Per creare il IAM ruolo, puoi adattare le istruzioni nel seguente tutorial,???.

c. Per utilizzare un sistema di autorizzazione Lambda, seleziona un token o un sistema di autorizzazione di richiesta. Affinché questa scelta sia visualizzata nel menu a discesa, è necessario creare un sistema di autorizzazione Lambda. Per informazioni su come creare un'autorizzazione Lambda, consulta Usa gli API autorizzatori Gateway Lambda.

d. Per utilizzare un pool di utenti di Amazon Cognito, sceglierne uno disponibile in Cognito user pool authorizers (Autorizzazioni pool di utenti Cognito). Crea un pool di utenti in Amazon Cognito e un autorizzatore del pool di utenti Amazon Cognito API in Gateway per visualizzare questa scelta nel menu a discesa. Per informazioni su come creare un'autorizzazione del pool di utenti di Amazon Cognito, consulta Controlla l'accesso alle API REST utilizzando i pool di utenti di Amazon Cognito come autorizzatore.

4. Per specificare la convalida della richiesta, seleziona un valore dal menu a discesa Validatore richiesta. Per disattivare la convalida della richiesta, seleziona Nessuno. Per ulteriori informazioni su ciascuna opzione, consulta Richiedi la convalida per REST APIs in API Gateway.

5. Seleziona la APIchiave richiesta per richiedere una chiave. API Se abilitate, API le chiavi vengono utilizzate nei piani di utilizzo per limitare il traffico dei clienti.

6. (Facoltativo) Per assegnare un nome SDK di operazione in un Java generato da API GatewayAPI, in Nome operazione, immettete un nome. Ad esempio, per la richiesta del metodo di`GET /pets/{petId}`, il nome dell'SDKoperazione Java corrispondente è, per impostazione predefinita,`GetPetsPetId`. Questo nome è costituito dal HTTP verbo (`GET`) del metodo e dai nomi delle variabili del percorso della risorsa (`PetsandPetId`). Se impostate il nome dell'operazione come`getPetById`, il nome dell'SDKoperazione diventa`GetPetById`.

7. Per aggiungere un parametro stringa di query al metodo, procedi come indicato di seguito:

   a. Scegliete Parametri della stringa di URL query, quindi scegliete Aggiungi stringa di query.

b. In Nome digita il nome del parametro della stringa di query.

c. Se il parametro della stringa di query appena creato viene usato per la convalida della richiesta, scegli l'opzione Obbligatorio. Per ulteriori informazioni sulla convalida della richiesta, consulta Richiedi la convalida per REST APIs in API Gateway.

d. Se il parametro della stringa di query appena creato viene usato come parte di una chiave di caching, seleziona l'opzione Caching. Per ulteriori informazioni sul caching, consulta Uso dei parametri di metodo o di integrazione come chiavi di cache per indicizzare le risposte memorizzate nella cache.

Per rimuovere il parametro della stringa di query, scegli Rimuovi.

8. Per aggiungere un parametro di intestazione al metodo, procedi come indicato di seguito:

a. Scegliete le intestazioni della HTTP richiesta, quindi scegliete Aggiungi intestazione.

b. Per Nome immetti il nome dell'intestazione.

c. Se l'intestazione appena creata viene usata per la convalida della richiesta, scegli l'opzione Obbligatorio. Per ulteriori informazioni sulla convalida della richiesta, consulta Richiedi la convalida per REST APIs in API Gateway.

d. Se l'intestazione appena creata viene usata come parte di una chiave di caching, scegli l'opzione Caching. Per ulteriori informazioni sul caching, consulta Uso dei parametri di metodo o di integrazione come chiavi di cache per indicizzare le risposte memorizzate nella cache.

Per rimuovere l'intestazione, scegli Rimuovi.

9. Per dichiarare il formato del payload di una richiesta di metodo con il `PATCH` HTTP verbo, o `POSTPUT`, scegliete Request body ed effettuate le seguenti operazioni:

a. Scegliere Add model (Aggiungi modello).

b. Per Content-type, inserite un MIME -type (ad esempio,). `application/json`

c. Per Modello seleziona un modello dal menu a discesa. I modelli attualmente disponibili per la API includono i modelli di default `Empty` e `Error` i modelli, nonché tutti i modelli che avete creato e aggiunto alla collezione Models di. API Per ulteriori informazioni sulla creazione di un modello, consulta Modelli di dati per REST APIs.

> **Note**
>
> Il modello è utile per informare il client del formato dati previsto di un payload, nonché per generare un modello di mappatura. È importante generare un codice fortemente tipizzato SDK API in linguaggi come Java, C#, Objective-C e Swift. È obbligatorio solo se la convalida di richiesta è abilitata per il payload.

10. Seleziona Salva.

Imposta una risposta del metodo API Gateway utilizzando la console Gateway API

Un API metodo può avere una o più risposte. Ogni risposta è indicizzata in base al relativo codice di HTTP stato. Per impostazione predefinita, la console API Gateway aggiunge `200` una risposta alle risposte del metodo. Puoi modificarla in modo che il metodo restituisca `201`. Puoi aggiungere altre risposte, ad esempio `409` per negare l'accesso e `500` per le variabili di fase non inizializzate utilizzate.

Per utilizzare la console API Gateway per modificare, eliminare o aggiungere una risposta a un API metodo, segui queste istruzioni.

1. Nel riquadro Risorse, scegli il metodo, quindi scegli la scheda Metodo di risposta. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. Nella sezione Impostazioni risposta metodo scegli Crea risposta.
3. Per il HTTPcodice di HTTP stato, inserisci un codice di stato come `200``400`, o`500`.

   Quando per una risposta restituita dal backend non è definito un metodo di risposta corrispondente, API Gateway non riesce a restituire la risposta al client. Restituisce invece una risposta di errore `500 Internal server error`.
4. Seleziona Add header (Aggiungi intestazione).
5. Per Nome intestazione immetti un nome.

   Per restituire un'intestazione dal back-end al client, aggiungi l'intestazione nella risposta del metodo.
6. Scegli Aggiungi modello per definire un formato del corpo della risposta del metodo.

   Immetti il tipo di supporto del payload della risposta per Tipo di contenuto e seleziona un modello dal menu a discesa Modelli.

7. Seleziona Salva.

Per modificare una risposta esistente, passa alla risposta del metodo e scegli Modifica. Per modificare il codice di HTTP stato, scegli Elimina e crea un nuovo metodo di risposta.

Per ogni risposta restituita dal back-end, deve essere configurata una risposta compatibile come risposta del metodo. Tuttavia le intestazioni delle risposte del metodo di configurazione e il modello di payload sono facoltativi, a meno che non si mappi il risultato dal back-end alla risposta del metodo prima della restituzione al client. Inoltre, un modello di payload di risposta al metodo è importante se state generando un payload fortemente tipizzato SDK per il vostro. API

Controlla e gestisci l'accesso alle API REST in API Gateway

API Gateway supporta più meccanismi per controllare e gestire l'accesso all'API.

Puoi usare i seguenti meccanismi per l'autenticazione e l'autorizzazione:

- Le policy delle risorse consentono di creare policy basate sulle risorse per permettere o negare l'accesso alle API e ai metodi da indirizzi IP o endpoint VPC di origine specificati. Per ulteriori informazioni, consulta the section called “Utilizza le politiche delle risorse del API Gateway”.
- I ruoli e le policy AWS IAM standard offrono controlli di accesso flessibili e robusti che possono essere applicati a un'intera API o a singoli metodi. Puoi utilizzare ruoli e policy IAM per controllare chi può creare, gestire e richiamare le API. Per ulteriori informazioni, consulta the section called “Usa le IAM autorizzazioni”.
- I tag IAM possono essere utilizzati insieme alle policy IAM per controllare l'accesso. Per ulteriori informazioni, consulta the section called “Controllo dell'accesso basato sugli attributi”.
- Le policy endpoint per endpoint VPC di interfaccia consentono di collegare le policy delle risorse IAM agli endpoint VPC di interfaccia per migliorare la sicurezza delle API private. Per ulteriori informazioni, consulta the section called “Usa le policy VPC degli endpoint per uso privato APIs”.
- Le autorizzazioni Lambda sono funzioni Lambda che controllano l'accesso ai metodi API REST usando l'autenticazione con token di connessione, nonché le informazioni descritte da intestazioni, percorsi, stringhe di query, variabili di fase o parametri di richiesta di variabili di contesto. Le autorizzazioni Lambda vengono utilizzate per controllare chi può richiamare i metodi API REST. Per ulteriori informazioni, consulta the section called “Uso di autorizzazioni Lambda”.
- I pool di utenti di Amazon Cognito consentono di creare soluzioni di autenticazione e autorizzazione personalizzabili per le API REST. I pool di utenti di Amazon Cognito vengono utilizzati per controllare chi può richiamare i metodi delle API REST. Per ulteriori informazioni,

consulta the section called “Utilizza un pool di utenti di Amazon Cognito come autorizzazione per un'API REST”.

Puoi utilizzare i seguenti meccanismi per l'esecuzione di altre attività correlate al controllo degli accessi:

- La funzionalità Cross-origin resource sharing (CORS) consente di controllare il modo in cui l'API REST risponde alle richieste di risorse multidominio. Per ulteriori informazioni, consulta the section called “CORS”.
- I certificati SSL lato client possono essere utilizzati per verificare che le richieste HTTP al sistema back-end provengano da API Gateway. Per ulteriori informazioni, consulta the section called “Certificati del client”.
- AWS WAF può essere utilizzato per proteggere l'API di API Gateway da exploit di Web comuni. Per ulteriori informazioni, consulta the section called “AWS WAF”.

Puoi utilizzare i seguenti meccanismi per monitorare e limitare l'accesso che è stato consentito ai client autorizzati:

- I piani di utilizzo consentono di fornire chiavi API ai clienti e quindi di monitorare e limitare l'utilizzo delle fasi e dei metodi API per ciascuna chiave API. Per ulteriori informazioni, consulta the section called “Piani di utilizzo”.

Controlla l'accesso a una politica delle risorse di REST API API With Gateway

Le policy relative alle risorse di Amazon API Gateway sono documenti di JSON policy allegati API a un account per controllare se un responsabile specifico (in genere un IAM ruolo o un gruppo) può richiamare ilAPI. Puoi utilizzare le policy relative alle risorse di API Gateway per API consentirti di essere invocato in modo sicuro da:

- Utenti di un account specificato AWS .
- Intervalli o CIDR blocchi di indirizzi IP di origine specificati.
- Cloud privati virtuali (VPCs) o VPC endpoint (in qualsiasi account).

È possibile allegare una politica delle risorse per qualsiasi tipo di API endpoint in API Gateway utilizzando AWS Management Console AWS CLI, o. AWS SDKs In ambito privato APIs, è possibile utilizzare le politiche delle risorse insieme alle politiche VPC degli endpoint per controllare quali

principali hanno accesso a quali risorse e azioni. Per ulteriori informazioni, consulta the section called "Usa le policy VPC degli endpoint per uso privato APIs".

APILe policy relative alle risorse del gateway sono diverse dalle politiche basate sull'IAMidentità. IAMle politiche basate sull'identità sono associate a IAM utenti, gruppi o ruoli e definiscono quali azioni tali identità sono in grado di eseguire su quali risorse. APILe policy relative alle risorse del gateway sono allegate alle risorse. È possibile utilizzare le politiche delle risorse del API Gateway insieme IAM alle politiche. Per ulteriori informazioni, consulta Policy basate sulle identità e policy basate su risorse.

Argomenti

- Panoramica linguistica delle policy di accesso per Amazon API Gateway
- In che modo le politiche delle risorse del API gateway influiscono sul flusso di lavoro
- APIEsempi di policy sulle risorse di Gateway
- Creare e allegare una politica delle risorse API Gateway a un API
- AWS chiavi di condizione che possono essere utilizzate nelle politiche relative alle risorse di API Gateway

Panoramica linguistica delle policy di accesso per Amazon API Gateway

Questa pagina descrive gli elementi di base utilizzati nelle politiche delle risorse di Amazon API Gateway.

Le politiche relative alle risorse vengono specificate utilizzando la stessa sintassi delle IAM politiche. Per informazioni complete sul linguaggio delle politiche, vedere Panoramica delle IAM politiche e Riferimento alle AWS Identity and Access Management politiche nella Guida per l'IAMutente.

Per informazioni su come un AWS servizio decide se una determinata richiesta deve essere consentita o rifiutata, vedere Determinare se una richiesta è consentita o rifiutata.

Elementi comuni in una policy di accesso

In termini basilari, una policy delle risorse contiene i seguenti elementi:

- Risorse: APIs sono le risorse Amazon API Gateway per le quali puoi consentire o negare le autorizzazioni. In una policy, usi Amazon Resource Name (ARN) per identificare la risorsa. Puoi anche utilizzare una sintassi abbreviata, che API Gateway espande completamente automaticamente ARN quando salvi una politica sulle risorse. Per ulteriori informazioni, consulta APIEsempi di policy sulle risorse di Gateway.

Per il formato dell'elemento `Resource` completo, consulta Formato delle risorse delle autorizzazioni per l'esecuzione in Gateway API API.

- Azioni: per ogni risorsa, Amazon API Gateway supporta una serie di operazioni. Vengono identificate le operazioni delle risorse che verranno consentite (o rifiutate) utilizzando le parole chiave dell'operazione.

  Ad esempio, l'`execute-api:Invoke`autorizzazione consentirà all'utente di richiamare un messaggio API su richiesta del cliente.

  Per il formato dell'elemento `Action`, consulta Formato di azione delle autorizzazioni per l'esecuzione API in Gateway API.
- Effetto: l'effetto prodotto quando l'utente richiede una determinata operazione; può essere `Allow` o `Deny`. È anche possibile rifiutare esplicitamente l'accesso a una risorsa per essere certi che un utente non sia in grado di accedervi, anche quando tale accesso è assegnato da un'altra policy.

  > **Note**
  >
  > "Rifiuto implicito" ha lo stesso significato di "rifiuto per default".
  > Un "rifiuto implicito" è diverso da un "rifiuto esplicito". Per ulteriori informazioni, consulta la sezione sulla differenza tra rifiuto per default e rifiuto esplicito.
- Principale: account o utente autorizzato ad accedere alle operazioni e alle risorse nell'istruzione. In una policy delle risorse, il principale è l'utente o l'account che riceve questa autorizzazione.

L'esempio di policy delle risorse seguente mostra gli elementi di policy comuni descritti in precedenza. La politica garantisce l'accesso a quanto API specificato *`account-id`* nel specificato *`region`* a qualsiasi utente il cui indirizzo IP di origine si trova nel blocco di indirizzi *`123.4.5.6/24`*. La politica nega qualsiasi accesso a API se l'IP di origine dell'utente non rientra nell'intervallo.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "arn:aws:execute-api:region:account-id:*"
        },
```

```
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "arn:aws:execute-api:region:account-id:*",
            "Condition": {
                "NotIpAddress": {
                    "aws:SourceIp": "123.4.5.6/24"
                }
            }
        }
    ]
}
```

In che modo le politiche delle risorse del API gateway influiscono sul flusso di lavoro

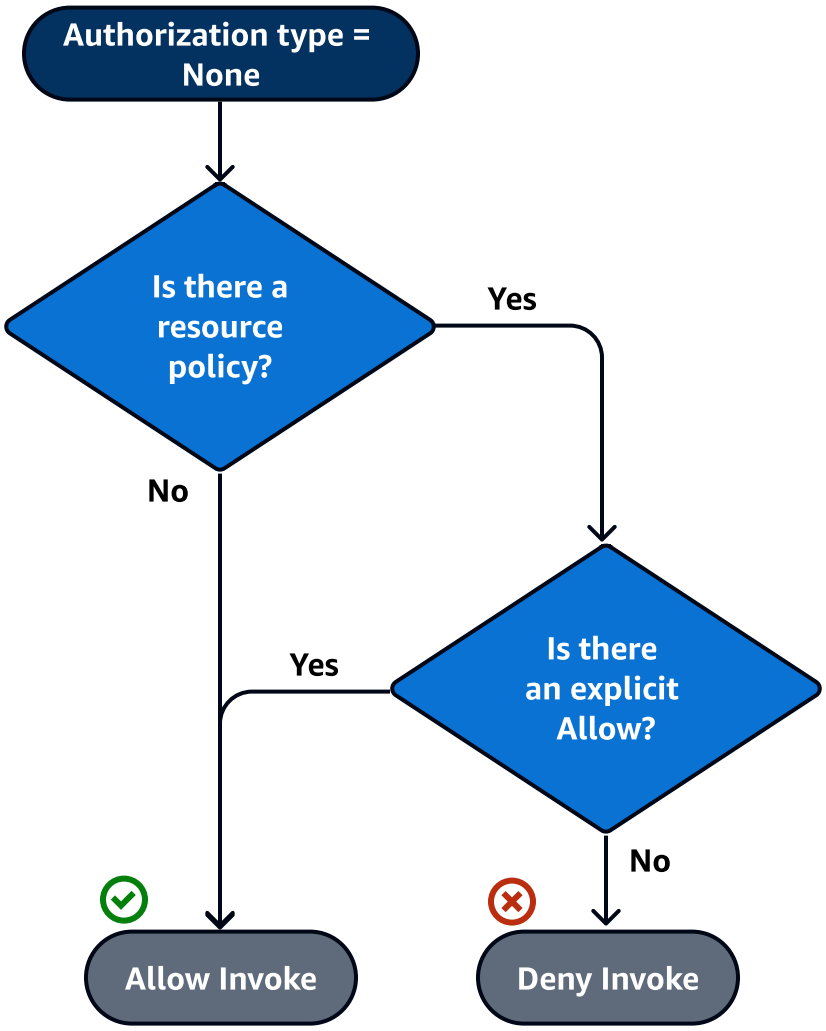
Quando API Gateway valuta la politica delle risorse allegata alla vostraAPI, il risultato è influenzato dal tipo di autenticazione che avete definito per laAPI, come illustrato nei diagrammi di flusso nelle sezioni seguenti.

Argomenti

- APISolo policy relative alle risorse di Gateway
- Autorizzazione Lambda e policy delle risorse
- Autenticazione IAM e policy delle risorse
- Autenticazione Amazon Cognito e policy delle risorse
- Tabelle dei risultati della valutazione delle policy

APISolo policy relative alle risorse di Gateway

In questo flusso di lavoro, una politica delle risorse del API Gateway è allegata aAPI, ma non è definito alcun tipo di autenticazione perAPI. La valutazione della policy comporta la ricerca di un'autorizzazione esplicita basata sui criteri in entrata dell'intermediario. Un rifiuto implicito o esplicito comporta il rifiuto dell'intermediario.



Di seguito viene riportato un esempio di tale policy delle risorse.

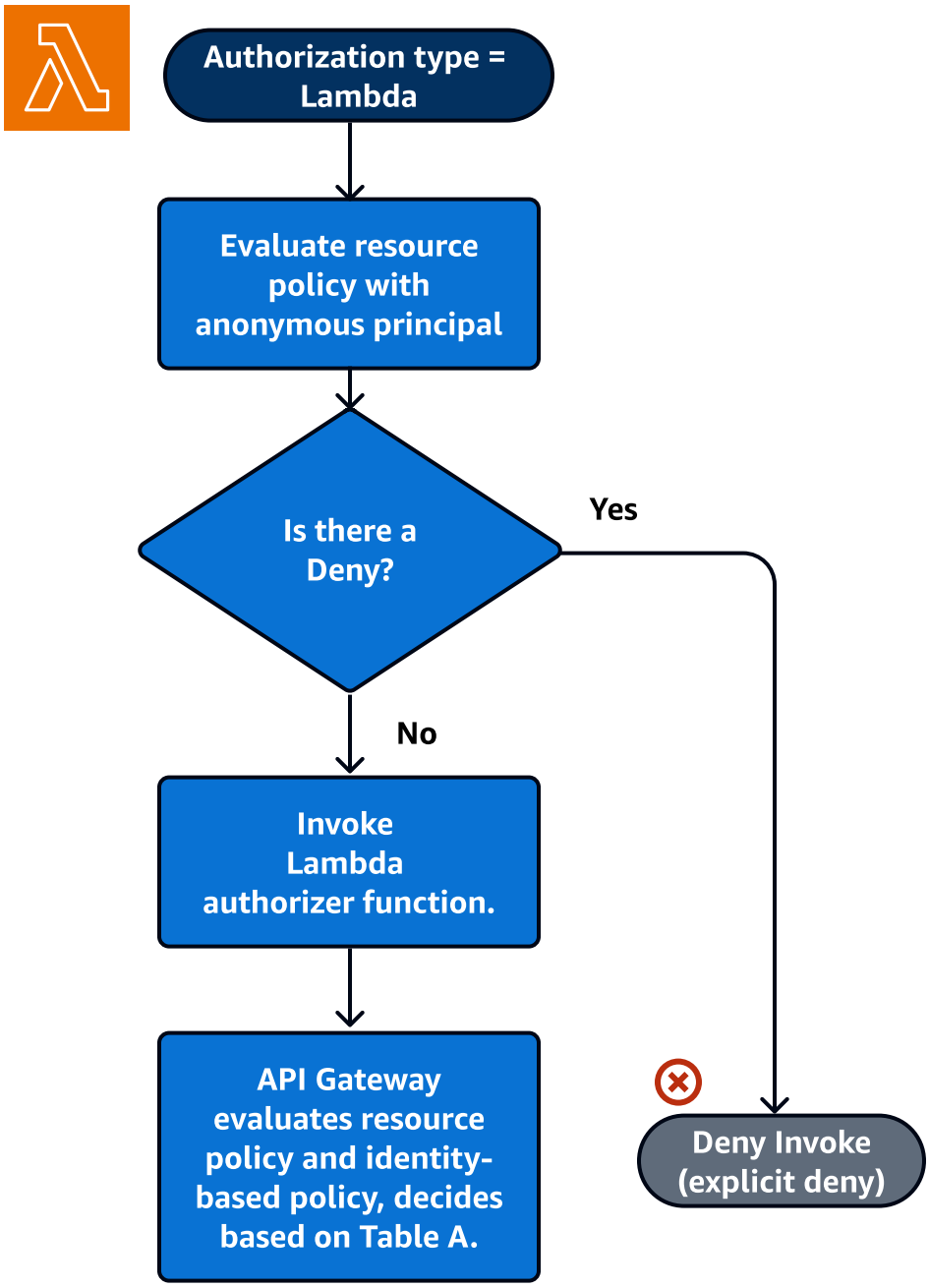
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "arn:aws:execute-api:region:account-id:api-id/",
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": ["192.0.2.0/24", "198.51.100.0/24" ]
                }
            }
```

```
        }
    ]
}
```

Autorizzazione Lambda e policy delle risorse

In questo flusso di lavoro, viene configurato un autorizzatore Lambda per la politica delle risorse API in aggiunta a quella delle risorse. La policy delle risorse viene valutata in due fasi. Prima di chiamare l'autorizzatore Lambda, API Gateway valuta innanzitutto la policy e verifica eventuali negazioni esplicite. Se riscontrati, all'intermediario viene negato subito l'accesso. In caso contrario viene chiamata l'autorizzazione Lambda, che restituisce un documento della policy che viene valutato insieme alla policy delle risorse. Il risultato è determinato in base alla Tabella A.

La politica delle risorse di esempio seguente consente le chiamate solo dall'VPCendpoint il cui ID VPC endpoint è. `vpce-1a2b3c4d` Durante la valutazione «pre-auth», solo le chiamate provenienti dall'VPCendpoint indicato nell'esempio possono andare avanti e valutare l'autorizzatore Lambda. Tutte le chiamate rimanenti vengono bloccate.




```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
```

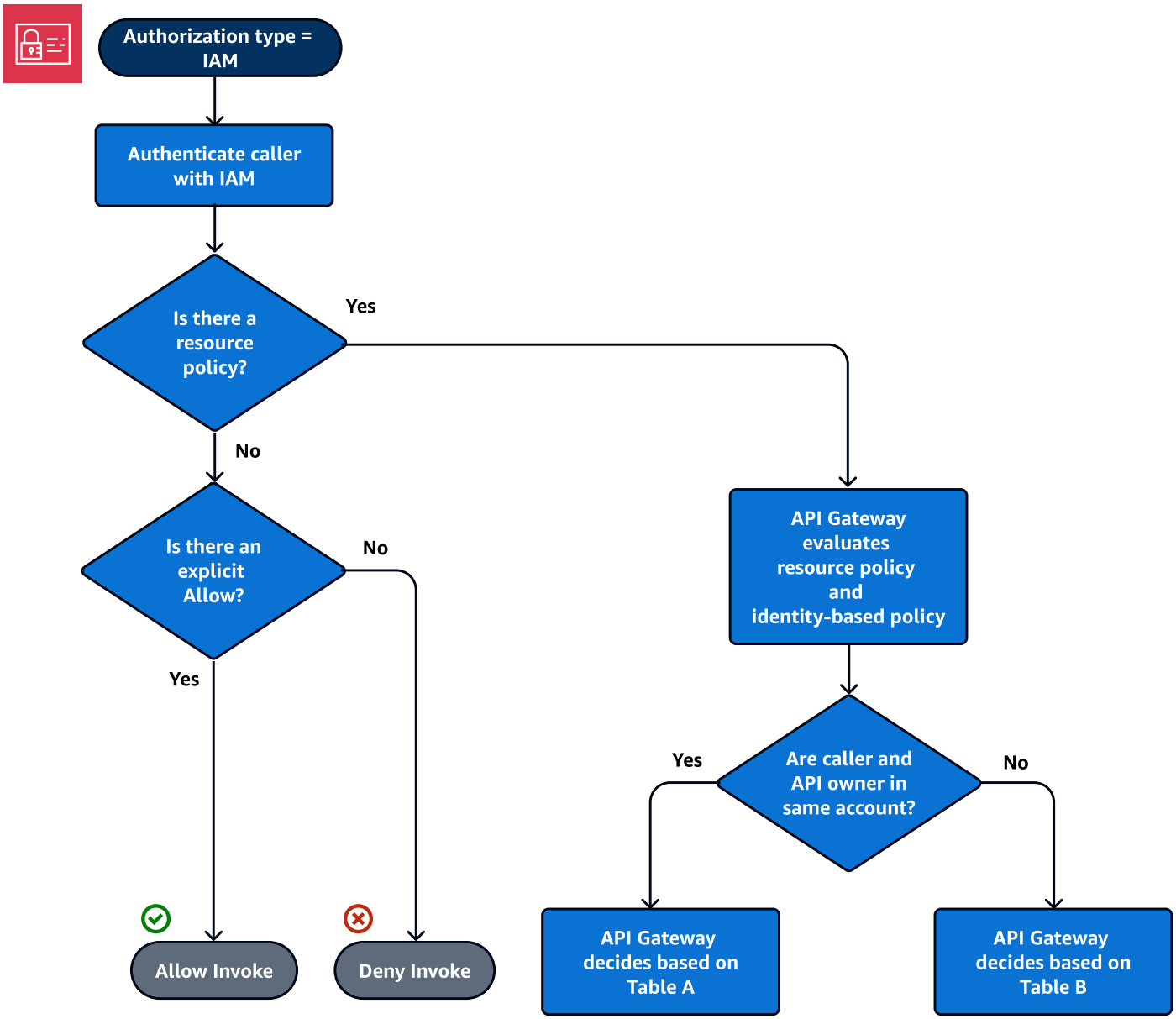
```
                "arn:aws:execute-api:region:account-id:api-id/"
            ],
            "Condition" : {
                "StringNotEquals": {
                    "aws:SourceVpce": "vpce-1a2b3c4d"
                }
            }
        }
    ]
}
```

Autenticazione IAM e policy delle risorse

In questo flusso di lavoro, si configura IAM l'autenticazione per oltre API a una politica delle risorse. Dopo aver autenticato l'utente con il IAM servizio, API valuta sia le politiche allegate all'utente che la politica delle risorse. Il risultato varia a seconda che il chiamante si trovi nella stessa sede Account AWS o in una posizione diversa dal Account AWS proprietario. API

Se il chiamante e il API proprietario provengono da account separati, sia i criteri che i IAM criteri relativi alle risorse consentono esplicitamente al chiamante di procedere. Per ulteriori informazioni, vedere la Tabella B.

Tuttavia, se il chiamante e il API proprietario coincidono Account AWS, le politiche IAM degli utenti o le politiche delle risorse devono consentire esplicitamente al chiamante di procedere. Per ulteriori informazioni, vedere la Tabella A.



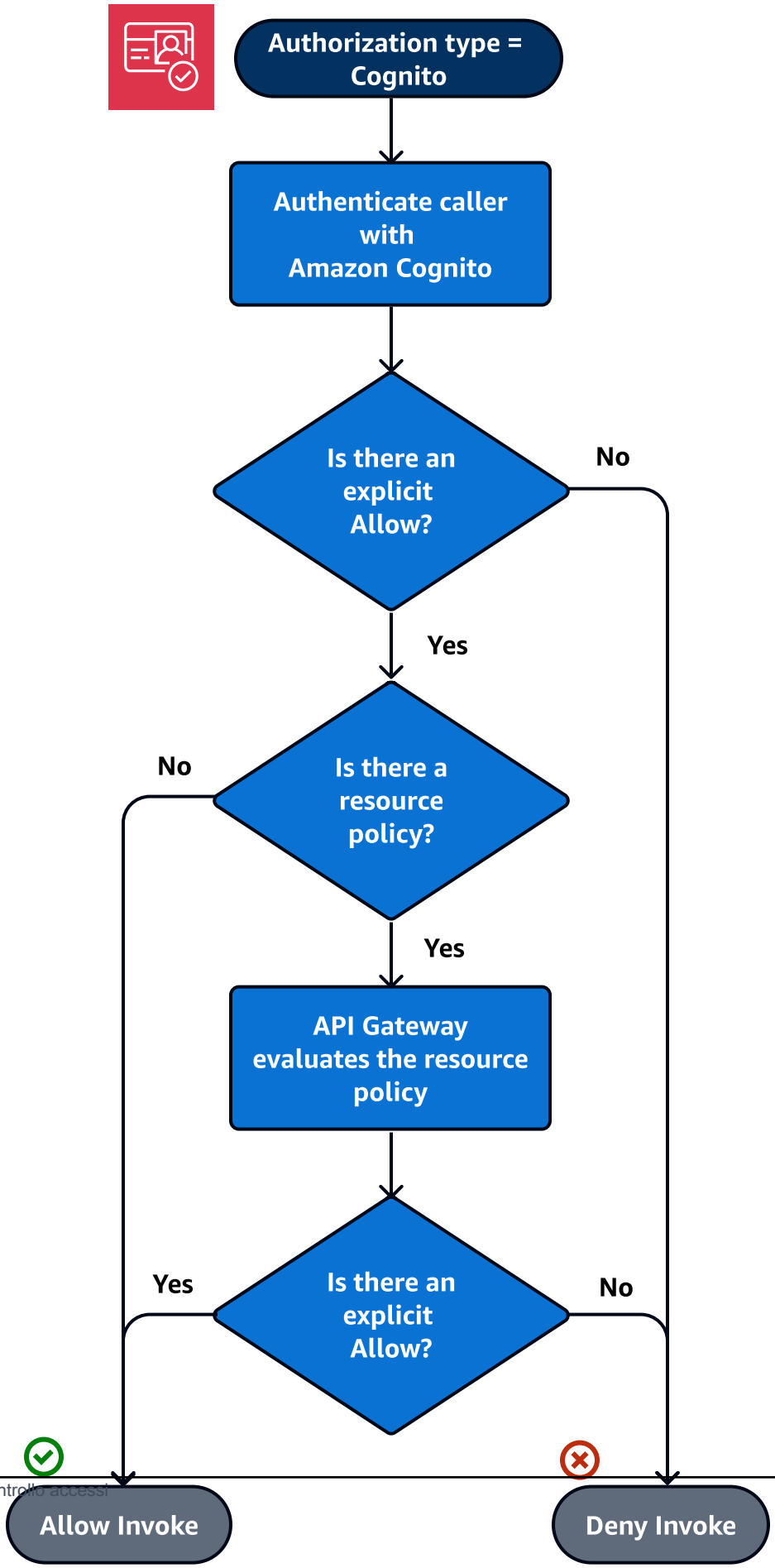
Di seguito viene riportato un esempio di policy delle risorse multiaccount. Supponendo che la IAM politica contenga un effetto di autorizzazione, questa politica delle risorse consente solo le chiamate provenienti dal VPC cui VPC ID è*vpc-2f09a348*. Per ulteriori informazioni, vedere la Tabella B.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
```

```
            "Resource": [
                "arn:aws:execute-api:region:account-id:api-id/"
            ],
            "Condition" : {
                "StringEquals": {
                    "aws:SourceVpc": "vpc-2f09a348"
                    }
            }
        }
    ]
}
```

Autenticazione Amazon Cognito e policy delle risorse

In questo flusso di lavoro, un pool di utenti di Amazon Cognito è configurato per la politica delle risorse API in aggiunta a quella delle risorse. APIGateway tenta innanzitutto di autenticare il chiamante tramite Amazon Cognito. Questa operazione viene in genere eseguita tramite un JWTtoken fornito dal chiamante. Se l'autenticazione va a buon fine, la policy delle risorse viene valutata in modo indipendente ed è richiesta un'autorizzazione esplicita. Un rifiuto o "non consentire né rifiutare" si traduce in un rifiuto. Di seguito è riportato un esempio di una policy delle risorse che potrebbe essere utilizzata insieme a pool di utenti di Amazon Cognito.


Authorization type = Cognito
Authenticate caller with Amazon Cognito
Is there an explicit Allow?
No
Yes
Is there a resource policy?
No
Yes
API Gateway evaluates the resource policy
Is there an explicit Allow?
Yes
No
Allow Invoke
Deny Invoke

Di seguito è riportato un esempio di politica delle risorse che consente chiamate solo da una fonte specificataIPs, presupponendo che il token di autenticazione Amazon Cognito contenga un'autorizzazione. Per ulteriori informazioni, consulta la Tabella B.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "arn:aws:execute-api:region:account-id:api-id/",
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": ["192.0.2.0/24", "198.51.100.0/24" ]
                }
            }
        }
    ]
}
```

Tabelle dei risultati della valutazione delle policy

La Tabella A elenca il comportamento risultante quando l'accesso a un API Gateway API è controllato da una IAM policy o da un'autorizzazione Lambda e da una policy delle risorse API Gateway, entrambe coincidenti. Account AWS

IAMpolicy (o autorizzatore Lambda)	APIPolitica delle risorse del gateway	Comportamento risultante
Abilita	Abilita	Abilita
Abilita	Né Abilita né Rifiuta	Abilita
Abilita	Rifiuta	Explicit Deny (rifiuto esplicito)
Né Abilita né Rifiuta	Abilita	Abilita
Né Abilita né Rifiuta	Né Abilita né Rifiuta	Rifiuto implicito
Né Abilita né Rifiuta	Rifiuta	Explicit Deny (rifiuto esplicito)

IAMpolicy (o autorizzatore Lambda)	APIPolitica delle risorse del gateway	Comportamento risultante
Rifiuta	Abilita	Explicit Deny (rifiuto esplicito)
Rifiuta	Né Abilita né Rifiuta	Explicit Deny (rifiuto esplicito)
Rifiuta	Rifiuta	Explicit Deny (rifiuto esplicito)

La Tabella B elenca il comportamento risultante quando l'accesso a un API Gateway API è controllato da una IAM policy o da un autorizzatore di pool di utenti di Amazon Cognito e da una policy di risorse API Gateway, che sono diverse. Account AWS Se una delle due è silente (non permette né rifiuta). l'accesso multiaccount viene negato. Questo perché l'accesso tra account diversi richiede che sia la politica delle risorse che la IAM policy o l'autorizzatore dei pool di utenti di Amazon Cognito concedano esplicitamente l'accesso.

IAMpolicy (o autorizzatore dei pool di utenti di Amazon Cognito)	APIPolitica delle risorse di Gateway	Comportamento risultante
Abilita	Abilita	Abilita
Abilita	Né Abilita né Rifiuta	Rifiuto implicito
Abilita	Rifiuta	Explicit Deny (rifiuto esplicito)
Né Abilita né Rifiuta	Abilita	Rifiuto implicito
Né Abilita né Rifiuta	Né Abilita né Rifiuta	Rifiuto implicito
Né Abilita né Rifiuta	Rifiuta	Explicit Deny (rifiuto esplicito)
Rifiuta	Abilita	Explicit Deny (rifiuto esplicito)
Rifiuta	Né Abilita né Rifiuta	Explicit Deny (rifiuto esplicito)
Rifiuta	Rifiuta	Explicit Deny (rifiuto esplicito)

APIEsempi di policy sulle risorse di Gateway

Questa pagina presenta alcuni esempi di casi d'uso tipici delle policy relative alle risorse di API Gateway.

Le politiche di esempio seguenti utilizzano una sintassi semplificata per specificare la API risorsa. Questa sintassi semplificata è un modo abbreviato per fare riferimento a una API risorsa, anziché specificare l'intero Amazon Resource Name (). ARN APIGateway converte la sintassi abbreviata nella versione completa quando salvi la policy. ARN Ad esempio, è possibile specificare la risorsa `execute-api:/stage-name/GET/pets` in una policy delle risorse. APIGateway converte la risorsa in `arn:aws:execute-api:us-east-2:123456789012:aabbccddee/stage-name/GET/pets` quando si salva la politica delle risorse. APIGateway crea il ARN file completo utilizzando la regione corrente, l'ID AWS dell'account e l'ID a REST API cui è associata la politica delle risorse. È possibile utilizzare `execute-api:/*` per rappresentare tutte le fasi, i metodi e i percorsi della versione correnteAPI. Per informazioni sulla sintassi della/e policy di accesso, consulta Panoramica linguistica delle policy di accesso per Amazon API Gateway.

Argomenti

- Esempio: consenti ai ruoli di un altro AWS account di utilizzare un API
- Esempio: nega API il traffico in base all'indirizzo o all'intervallo IP di origine
- Esempio: nega il API traffico in base all'indirizzo o all'intervallo IP di origine quando si utilizza un indirizzo privato API
- Esempio: consenti il API traffico privato in base alla sorgente o all'endpoint VPC VPC

Esempio: consenti ai ruoli di un altro AWS account di utilizzare un API

L'esempio seguente di politica delle risorse concede API l'accesso in un AWS account a due ruoli in un AWS account diverso tramite i protocolli Signature Version 4 (SigV4). In particolare, allo sviluppatore e al ruolo di amministratore dell' AWS account identificato da *account-id-2* viene concessa l'`execute-api:Invoke`azione per eseguire l'`GET`azione sulla `pets` risorsa (API) dell'account. AWS

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
            "Effect": "Allow",
```

```
            "Principal": {
                "AWS": [
                    "arn:aws:iam::account-id-2:role/developer",
                    "arn:aws:iam::account-id-2:role/Admin"
                ]
            },
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/stage/GET/pets"
            ]
        }
    ]
}
```

Esempio: nega API il traffico in base all'indirizzo o all'intervallo IP di origine

Il seguente esempio di policy in materia di risorse nega (blocca) il traffico in entrata API da e verso due blocchi di indirizzi IP di origine specificati.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
               "execute-api:/*"
            ],
            "Condition" : {
                "IpAddress": {
                    "aws:SourceIp": ["192.0.2.0/24", "198.51.100.0/24" ]
                }
            }
        }
```

```
    ]
}
```

Esempio: nega il API traffico in base all'indirizzo o all'intervallo IP di origine quando si utilizza un indirizzo privato API

L'esempio seguente di policy in materia di risorse nega (blocca) il traffico in entrata verso un privato proveniente API da due blocchi di indirizzi IP di origine specificati. Quando si utilizza privateAPIs, l'VPCendpoint for `execute-api` riscrive l'indirizzo IP di origine originale. La condizione `aws:VpcSourceIp` filtra la richiesta rispetto all'indirizzo IP del richiedente originale.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
               "execute-api:/*"
            ],
            "Condition" : {
                "IpAddress": {
                    "aws:VpcSourceIp": ["192.0.2.0/24", "198.51.100.0/24"]
                }
            }
        }
    ]
}
```

Esempio: consenti il API traffico privato in base alla sorgente o all'endpoint VPC VPC

L'esempio seguente di policy relative alle risorse consente il traffico in entrata verso un ambiente privato API solo da un cloud privato virtuale (VPC) o VPC da un endpoint specifico.

Questo esempio di politica delle risorse specifica una fonte: VPC

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ],
            "Condition" : {
                "StringNotEquals": {
                   "aws:SourceVpc": "vpc-1a2b3c4d"
                }
            }
        }
    ]
}
```

Questo esempio di politica delle risorse specifica un endpoint di origine: VPC

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
```

```
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ],
            "Condition" : {
                "StringNotEquals": {
                    "aws:SourceVpce": "vpce-1a2b3c4d"
                }
            }
        }
    ]
}
```

Creare e allegare una politica delle risorse API Gateway a un API

Per consentire a un utente di accedere al servizio di esecuzione API chiamando il servizio di API esecuzione, è necessario creare una politica delle risorse API Gateway e allegare la politica aAPI. Quando si allega una politica alla propriaAPI, vengono applicate le autorizzazioni contenute nella politica ai metodi in. API Se aggiorni la politica delle risorse, dovrai implementare il. API

Argomenti

- Prerequisiti
- Allega una politica delle risorse a un API Gateway API
- Risolvi i problemi relativi alla politica delle risorse

Prerequisiti

Per aggiornare una politica delle risorse di API Gateway, avrai bisogno dell'`apigateway:UpdateRestApiPolicy`autorizzazione e dell'`apigateway:PATCH`autorizzazione.

Nel caso di una politica regionale o ottimizzata per i dispositivi perifericiAPI, puoi allegare la politica delle risorse al API momento della creazione o dopo che è stata implementata. Se si tratta di una politica privataAPI, non è possibile implementarla senza una politica in materia di risorseAPI. Per ulteriori informazioni, consulta the section called “Privato REST APIs”.

Allega una politica delle risorse a un API Gateway API

La procedura seguente mostra come allegare una politica delle risorse a un API GatewayAPI.

AWS Management Console

Per allegare una politica delle risorse a un API Gateway API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli un RESTAPI.
3. Nel riquadro di navigazione principale, scegli Policy delle risorse.
4. Scegli Crea policy.
5. (Facoltativo) Scegli Seleziona un modello per generare una policy di esempio.

   Negli esempi di policy, i segnaposto sono racchiusi tra parentesi graffe doppie ("{{*placeholder*}}"). Sostituisci ogni segnaposto, incluse le parentesi graffe, con le informazioni necessarie.
6. Se non utilizzi uno dei modelli di esmepio, immetti la tua policy delle risorse.
7. Seleziona Salvataggio delle modifiche.

Se API è stato distribuito in precedenza nella console API Gateway, sarà necessario ridistribuirlo affinché la politica delle risorse abbia effetto.

AWS CLI

Per utilizzare la AWS CLI per creare una nuova politica delle risorse API e allegarla, chiama il create-rest-apicomando come segue:

```
aws apigateway create-rest-api \
    --name "api-name" \
    --policy "{\"jsonEscapedPolicyDocument\"}"
```

Per utilizzare il AWS CLI per allegare una politica delle risorse a una esistenteAPI, chiamate il update-rest-apicomando come segue:

```
aws apigateway update-rest-api \
    --rest-api-id api-id \
    --patch-operations op=replace,path=/
policy,value='"{\"jsonEscapedPolicyDocument\"}"'
```

AWS CloudFormation

È possibile utilizzare AWS CloudFormation per creare una politica API con una risorsa. L'esempio seguente crea una politica delle risorse REST API con l'esempio,the section called “Esempio: nega API il traffico in base all'indirizzo o all'intervallo IP di origine”.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Name: testapi
      Policy:
        Statement:
          - Action: 'execute-api:Invoke'
            Effect: Allow
            Principal: '*'
            Resource: 'execute-api/*'
          - Action: 'execute-api:Invoke'
            Effect: Deny
            Principal: '*'
            Resource: 'execute-api/*'
            Condition:
              IpAddress:
                'aws:SourceIp': ["192.0.2.0/24", "198.51.100.0/24" ]
        Version: 2012-10-17
  Resource:
    Type: 'AWS::ApiGateway::Resource'
    Properties:
      RestApiId: !Ref Api
      ParentId: !GetAtt Api.RootResourceId
      PathPart: 'helloworld'
  MethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref Resource
      HttpMethod: GET
      ApiKeyRequired: false
      AuthorizationType: NONE
      Integration:
        Type: MOCK
  ApiDeployment:
```

```
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn:
      - MethodGet
    Properties:
      RestApiId: !Ref Api
      StageName: test
```

Risolvi i problemi relativi alla politica delle risorse

La seguente guida per la risoluzione dei problemi potrebbe aiutarti a risolvere i problemi relativi alla politica delle risorse.

My API return {"Message» :"User: anonymous is not authorized to perform: execute-api:invoke on resource: arn:aws:execute-api:us-east-1: ********/****/****/ "}

Nella tua politica delle risorse, AWS se imposti il Principal su un principale, come il seguente:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            ""Principal": {
                "AWS": [
                    "arn:aws:iam::account-id:role/developer",
                    "arn:aws:iam::account-id:role/Admin"
                ]
            },
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        ...
        }
```

È necessario utilizzare `AWS_IAM` l'autorizzazione per ogni metodo utilizzatoAPI, altrimenti verrà API restituito il messaggio di errore precedente. Per ulteriori istruzioni su come attivare `AWS_IAM` l'autorizzazione per un metodo, consultathe section called “Metodi”.

La mia politica sulle risorse non si aggiorna

Se aggiorni la politica delle risorse dopo API la creazione, dovrai implementare la politica API per propagare le modifiche dopo aver allegato la politica aggiornata. L'aggiornamento o il salvataggio della politica da soli non modificheranno il comportamento di runtime di. API Per ulteriori informazioni sulla distribuzione del tuoAPI, consultathe section called “Implementa REST APIs”.

La mia politica sulle risorse restituisce il seguente errore: Documento di policy non valido. Controlla la sintassi della politica e assicurati che i Principal siano validi.

Per risolvere questo errore, ti consigliamo innanzitutto di controllare la sintassi della policy. Per ulteriori informazioni, consulta the section called “Panoramica linguistica delle policy di accesso per Amazon API Gateway”. Ti consigliamo inoltre di verificare che tutti i principali specificati siano validi e non siano stati eliminati.

Inoltre, se ti trovi in una regione che accetta l'iscrizione, verifica che la regione API sia abilitata per tutti gli account indicati nella politica delle risorse.

AWS chiavi di condizione che possono essere utilizzate nelle politiche relative alle risorse di API Gateway

La tabella seguente contiene le chiavi di AWS condizione che possono essere utilizzate nelle politiche delle risorse di API Gateway per ogni tipo di autorizzazione. APIs

Per ulteriori informazioni sulle chiavi AWS condizionali, vedere AWS Global Condition Context Keys.

Chiavi condizionali	Criteri	Esigenze**`AuthN`**?	Tipo di autorizzazione
`aws:CurrentTime`	Nessuna	No	Tutti
`aws:EpochTime`	Nessuna	No	Tutti
`aws:Token IssueTime`	La chiave è presente solo nelle richieste firmate utilizzan do credenziali di sicurezza temporane e.	Sì	IAM

Chiavi condizionali	Criteri	Esigenze**AuthN**?	Tipo di autorizzazione
`aws:MultiFactorAuthPresent`	La chiave è presente solo nelle richieste firmate utilizzando credenziali di sicurezza temporanee.	Sì	IAM
`aws:MultiFactorAuthAge`	La chiave è presente solo se MFA è presente nelle richieste.	Sì	IAM
`aws:PrincipalAccount`	Nessuno	Sì	IAM
`aws:PrincipalArn`	Nessuno	Sì	IAM
`aws:PrincipalOrgID`	Questa chiave viene inclusa nel contesto della richiesta solo se l'entità è membro di un'organizzazione.	Sì	IAM
`aws:PrincipalOrgPaths`	Questa chiave viene inclusa nel contesto della richiesta solo se l'entità è membro di un'organizzazione.	Sì	IAM

Chiavi condizionali	Criteri	Esigenze**AuthN**?	Tipo di autorizzazione
`aws:PrincipalTag`	Questa chiave è inclusa nel contesto della richiesta se il principale utilizza un IAM utente con tag allegati. È inclusa per un principale che utilizza un IAM ruolo con tag allegati o tag di sessione.	Sì	IAM
`aws:PrincipalType`	Nessuno	Sì	IAM
`aws:Referer`	La chiave è presente solo se il valore è fornito dal chiamante nell'HTTPintestazione.	No	Tutti
`aws:SecureTransport`	Nessuna	No	Tutti
`aws:SourceArn`	Nessuna	No	Tutti
`aws:SourceIp`	Nessuna	No	Tutti
`aws:SourceVpc`	Questa chiave può essere utilizzata solo per uso privato. APIs	No	Tutti
`aws:SourceVpce`	Questa chiave può essere usata solo per uso privatoAPIs.	No	Tutti

Chiavi condizionali	Criteri	Esigenze**AuthN**?	Tipo di autorizzazione
`aws:VpcSourceIp`	Questa chiave può essere usata solo per uso privatoAPIs.	No	Tutti
`aws:UserAgent`	La chiave è presente solo se il valore è fornito dal chiamante nell'HTTPintestazione.	No	Tutti
`aws:userid`	Nessuna	Sì	IAM
`aws:username`	Nessuno	Sì	IAM

Controlla l'accesso a un REST API con IAM autorizzazioni

Puoi controllare l'accesso al tuo Amazon API Gateway API con IAMle autorizzazioni controllando l'accesso ai seguenti due processi componenti del API Gateway:

- Per creare, distribuire e gestire un API in API Gateway, devi concedere API allo sviluppatore le autorizzazioni necessarie per eseguire le azioni richieste supportate dal componente di API gestione di Gateway. API
- Per chiamare un dispositivo distribuito API o aggiornare la memorizzazione nella API cache, è necessario concedere al API chiamante le autorizzazioni per eseguire le IAM azioni richieste supportate dal componente di esecuzione di Gateway. API API

Il controllo degli accessi per i due processi prevede l'uso dei due diversi modelli di autorizzazione illustrati più avanti.

APIModello di autorizzazioni Gateway per la creazione e la gestione di un API

Per consentire a uno API sviluppatore di creare e gestire un API in API Gateway, è necessario creare politiche di IAM autorizzazione che consentano a API uno sviluppatore specifico di creare, aggiornare, distribuire, visualizzare o eliminare le entità richieste. API È possibile collegare la policy delle autorizzazioni a un utente, un ruolo o un gruppo.

Per fornire l'accesso, aggiungi autorizzazioni ai tuoi utenti, gruppi o ruoli:

- Utenti e gruppi in: AWS IAM Identity Center

  Crea un set di autorizzazioni. Segui le istruzioni riportate nella pagina Create a permission set (Creazione di un set di autorizzazioni) nella Guida per l'utente di AWS IAM Identity Center .

- Utenti gestiti IAM tramite un provider di identità:

  Crea un ruolo per la federazione delle identità. Segui le istruzioni riportate in Creazione di un ruolo per un provider di identità di terze parti (federazione) nella Guida per l'IAMutente.

- IAMutenti:
  - Crea un ruolo che l'utente possa assumere. Segui le istruzioni riportate nella sezione Creazione di un ruolo per un IAM utente nella Guida per l'IAMutente.
  - (Non consigliato) Collega una policy direttamente a un utente o aggiungi un utente a un gruppo di utenti. Segui le istruzioni riportate in Aggiungere autorizzazioni a un utente (console) nella Guida per l'IAMutente.

Per ulteriori informazioni su come usare il modello di autorizzazione, consulta the section called “APIPolitiche basate sull'identità di Gateway”.

APIModello di autorizzazioni Gateway per richiamare un API

Per consentire a un API chiamante di richiamare API o aggiornare la memorizzazione nella cache, è necessario creare IAM politiche che consentano a un API chiamante specificato di richiamare il metodo per il quale è abilitata l'APIautenticazione utente. Lo API sviluppatore imposta la `authorizationType` proprietà del metodo in modo che richieda che il chiamante invii le credenziali dell'utente `AWS_IAM` per l'autenticazione. Sarà quindi possibile collegare la policy all'utente, al gruppo o al ruolo.

In questa dichiarazione sulla politica IAM delle autorizzazioni, l'IAM`Resource`elemento contiene un elenco di API metodi distribuiti identificati da determinati HTTP verbi e percorsi di risorse Gateway. API L'IAM`Action`elemento contiene le azioni API Gateway API richieste per l'esecuzione. Queste azioni includono `execute-api:Invoke` o`execute-api:InvalidateCache`, dove `execute-api` designa il componente di API esecuzione sottostante di API Gateway.

Per ulteriori informazioni su come usare il modello di autorizzazione, consulta Controlla l'accesso per invocare un API.

Quando un API è integrato con un AWS servizio (ad esempio AWS Lambda) nel backend, API Gateway deve disporre anche delle autorizzazioni per accedere alle AWS risorse integrate (ad esempio, richiamando una funzione Lambda) per conto del chiamante. API Per concedere queste autorizzazioni, crea un IAM ruolo del servizio di tipo Gateway.AWS API Quando si crea questo ruolo nella console di IAM gestione, il ruolo risultante contiene la seguente politica di IAM attendibilità che dichiara API Gateway come entità attendibile autorizzata ad assumere il ruolo:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "apigateway.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Se si crea il IAM ruolo chiamando il comando create-role di CLI o un SDK metodo corrispondente, è necessario fornire la politica di fiducia di cui sopra come parametro di input di. `assume-role-policy-document` Non tentate di creare una politica di questo tipo direttamente nella console di IAM gestione o di chiamare il comando AWS CLI create-policy o un metodo corrispondente. SDK

APIAffinché Gateway possa richiamare il AWS servizio integrato, è inoltre necessario allegare a questo ruolo le politiche di IAM autorizzazione appropriate per la chiamata ai servizi integrati. AWS Ad esempio, per chiamare una funzione Lambda, è necessario includere la seguente politica di IAM autorizzazioni nel ruolo: IAM

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": "*"
        }
    ]
```

```
}
```

Lambda supporta le policy di accesso basate su risorse, che combinano policy di autorizzazione e di attendibilità. Quando si integra una funzione API con Lambda utilizzando API la console Gateway, non viene richiesto di impostare IAM questo ruolo in modo esplicito, poiché la console imposta automaticamente le autorizzazioni basate sulle risorse sulla funzione Lambda, con il consenso dell'utente.

> **Note**
>
> Per implementare il controllo dell'accesso a un AWS servizio, è possibile utilizzare il modello di autorizzazioni basato sul chiamante, in cui una politica di autorizzazioni è collegata direttamente all'utente o al gruppo del chiamante, oppure il modello di autorizzazione basato sui ruoli, in cui una politica di autorizzazioni è associata a un IAM ruolo che API Gateway può assumere. Le policy di autorizzazione possono differire nei due modelli. Ad esempio la policy basata su intermediario blocca l'accesso, mentre quella basata su ruolo lo consente. È possibile trarne vantaggio per richiedere che un utente acceda a un servizio solo tramite un Gateway. AWS API API

Controlla l'accesso per invocare un API

In questa sezione, imparerai a conoscere il modello di autorizzazioni per controllare l'accesso ai permessi di API utilizzoIAM. Mostriamo un modello di dichiarazione IAM politica e il riferimento alla dichiarazione politica. Il riferimento alla dichiarazione politica include i formati `Action` e `Resource` i campi relativi al servizio di API esecuzione. Utilizzate questi riferimenti per creare la vostra dichiarazione IAM politica. Quando si crea la dichiarazione IAM politica, potrebbe essere necessario considerare il modo in cui le politiche delle risorse di API Gateway influiscono sul flusso di lavoro di autorizzazione. Per ulteriori informazioni, consulta the section called “Come le policy delle risorse influiscono sul flusso di lavoro delle autorizzazioni”.

Per uso privatoAPIs, è necessario utilizzare una combinazione di una politica delle risorse di API Gateway e una politica VPC degli endpoint. Per ulteriori informazioni, consulta i seguenti argomenti:

- the section called “Utilizza le politiche delle risorse del API Gateway”
- the section called “Usa le policy VPC degli endpoint per uso privato APIs”

Controlla chi può chiamare un API metodo API Gateway con policy IAM

Per controllare chi può o non può chiamare un impiegato API con IAM autorizzazioni, crea un documento di IAM policy con le autorizzazioni richieste. Di seguito viene mostrato un modello per questo tipo di documento di policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Permission",
      "Action": [
        "execute-api:Execution-operation"
      ],
      "Resource": [
        "arn:aws:execute-api:region:account-id:api-id/stage/METHOD_HTTP_VERB/Resource-
path"
      ]
    }
  ]
}
```

Qui, *`Permission`* deve essere sostituito da `Allow` o `Deny` a seconda che si desideri concedere o revocare le autorizzazioni incluse. *`Execution-operation`*deve essere sostituito dalle operazioni supportate dal servizio di esecuzione. API *`METHOD_HTTP_VERB`*rappresenta un HTTP verbo supportato dalle risorse specificate. *`Resource-path`*è il segnaposto per il URL percorso di un'API`Resource`istanza distribuita che supporta detto. *`METHOD_HTTP_VERB`* Per ulteriori informazioni, consulta Dichiarazione di riferimento sulle IAM politiche per l'esecuzione API in API Gateway.

> **Note**
>
> IAMAffinché le politiche siano efficaci, è necessario aver abilitato IAM l'autenticazione sui API metodi impostando `AWS_IAM` la proprietà dei metodi. `authorizationType` In caso contrario, questi API metodi saranno accessibili al pubblico.

Ad esempio, per concedere a un utente l'autorizzazione a visualizzare un elenco di animali domestici esposti da uno specificoAPI, ma per negare all'utente l'autorizzazione ad aggiungere un animale domestico all'elenco, è possibile includere la seguente dichiarazione nella politica: IAM

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:account-id:api-id/*/GET/pets"
      ]
    },
    {
      "Effect": "Deny",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:account-id:api-id/*/POST/pets"
      ]
    }
  ]
}
```

Per concedere a un utente l'autorizzazione a visualizzare uno specifico animale domestico esposto da un API uomo configurato come`GET /pets/{petId}`, è possibile includere la seguente dichiarazione nella IAM politica:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:account-id:api-id/*/GET/pets/a1b2"
      ]
    }
  ]
}
```

Dichiarazione di riferimento sulle IAM politiche per l'esecuzione API in API Gateway

Le seguenti informazioni descrivono il formato Action and Resource delle IAM politiche relative alle autorizzazioni di accesso per l'esecuzione di unAPI.

Formato di azione delle autorizzazioni per l'esecuzione API in Gateway API

L'`Actionespressione` API -execution ha il seguente formato generale:

```
execute-api:action
```

dove *`action`* è un'azione di API esecuzione disponibile:

- *, che rappresenta tutte le operazioni che seguono.
- Invoke, usato per invocare un API su richiesta di un client.
- InvalidateCache, utilizzato per invalidare la API cache su richiesta di un client.

Formato delle risorse delle autorizzazioni per l'esecuzione in Gateway API API

L'`Resourceespressione` API -execution ha il seguente formato generale:

```
arn:aws:execute-api:region:account-id:api-id/stage-name/HTTP-VERB/resource-path-
specifier
```

dove:

- *`region`* è la AWS regione (come **`us-east-1`** o **`*`** per tutte le AWS regioni) che corrisponde a quella distribuita API per il metodo.
- *`account-id`* è l'ID dell' AWS account a 12 cifre del proprietario. REST API
- *`api-id`* è l'identificatore che API Gateway ha assegnato al metodoAPI.
- *`stage-name`* è il nome dello stadio associato al metodo.
- *`HTTP-VERB`* è il HTTP verbo che indica il metodo. Può essere uno dei seguenti:GET,,POST, PUTDELETE,PATCH.
- *`resource-path-specifier`* è il percorso del metodo desiderato.

> **Note**
>
> Se specifichi un carattere jolly (`*`), l'espressione `Resource` applica il carattere jolly al resto dell'espressione.

Alcuni esempi dell'espressione Resource includono:

- **`arn:aws:execute-api:*:*:*`**per qualsiasi percorso di risorse in qualsiasi fase, per qualsiasi API AWS area geografica.
- **`arn:aws:execute-api:us-east-1:*:*`**per qualsiasi percorso di risorse in qualsiasi fase, per qualsiasi percorso API nella AWS regione di`us-east-1`.
- **`arn:aws:execute-api:us-east-1:*:`*`api-id`*`/*`**per qualsiasi percorso di risorsa in qualsiasi fase, per il API con l'identificatore di *`api-id`* nella AWS regione us-east-1.
- **`arn:aws:execute-api:us-east-1:*:`*`api-id`*`/test/*`**per qualsiasi percorso di risorsa nella fase di`test`, per il API con l'identificatore di *`api-id`* nella AWS regione us-east-1.

Per ulteriori informazioni, consulta Documentazione di riferimento Amazon Resource Name (ARN) API Gateway.

IAMesempi di policy per i permessi di esecuzione API

Per informazioni sui modelli di autorizzazioni e per altre informazioni di base, consulta Controlla l'accesso per invocare un API.

La seguente dichiarazione politica fornisce all'utente il permesso di chiamare qualsiasi POST metodo lungo il percorso di`mydemoresource`, nella fase di`test`, for the API con l'identificatore di`a123456789`, supponendo che il corrispondente sia API stato distribuito nella regione us-east-1 AWS :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
```

```
      "Resource": [
        "arn:aws:execute-api:us-east-1:*:a123456789/test/POST/mydemoresource/*"
      ]
    }
  ]
}
```

La seguente dichiarazione politica di esempio fornisce all'utente il permesso di chiamare qualsiasi metodo sul percorso delle risorse`petstorewalkthrough/pets`, in qualsiasi fase, API con l'identificatore di`a123456789`, in qualsiasi AWS regione in cui è stato distribuito il corrispondente: API

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:*:*:a123456789/*/*/petstorewalkthrough/pets"
      ]
    }
  ]
}
```

Creare e collegare una policy a un utente

Per consentire a un utente di chiamare il servizio di API gestione o il servizio di API esecuzione, è necessario creare una IAM politica che controlli l'accesso alle entità API Gateway.

Per utilizzare l'editor delle JSON politiche per creare una politica

1. Accedi a AWS Management Console e apri la IAM console all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel riquadro di navigazione a sinistra, seleziona Policies (Policy).

   Se è la prima volta che selezioni Policy, verrà visualizzata la pagina Benvenuto nelle policy gestite. Seleziona Inizia.
3. Nella parte superiore della pagina, scegli Crea policy.

4. Nella sezione Policy editor, scegli l'JSONopzione.
5. Inserisci il seguente documento JSON di policy:

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "action-statement"
      ],
      "Resource" : [
        "resource-statement"
      ]
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "action-statement"
      ],
      "Resource" : [
        "resource-statement"
      ]
    }
  ]
}
```

6. Scegli Next (Successivo).

> **Note**
>
> È possibile passare dall'opzione Visual a quella dell'JSONeditor in qualsiasi momento. Tuttavia, se apporti modifiche o scegli Avanti nell'editor visuale, IAM potresti ristrutturare la tua politica per ottimizzarla per l'editor visuale. Per ulteriori informazioni, consulta la sezione Ristrutturazione delle politiche nella Guida per l'IAMutente.

7. Nella pagina Rivedi e crea, immettere un valore in Nome policy e Descrizione (facoltativo) per la policy in fase di creazione. Rivedi Autorizzazioni definite in questa policy per visualizzare le autorizzazioni concesse dalla policy.
8. Seleziona Crea policy per salvare la nuova policy.

In questa dichiarazione, sostituisci *`action-statement`* e *`resource-statement`* se necessario, e aggiungete altre istruzioni per specificare le entità API Gateway che desiderate consentire all'utente di gestire, i API metodi che l'utente può chiamare o entrambi. Per impostazione predefinita, l'utente non dispone di autorizzazioni, a meno che non sia presente un'esplicita istruzione `Allow` corrispondente.

Hai appena creato una IAM policy. Non avrà alcun effetto finché non viene collegato.

Per fornire l'accesso, aggiungi autorizzazioni ai tuoi utenti, gruppi o ruoli:

- Utenti e gruppi in AWS IAM Identity Center:

  Crea un set di autorizzazioni. Segui le istruzioni riportate nella pagina Create a permission set (Creazione di un set di autorizzazioni) nella Guida per l'utente di AWS IAM Identity Center .
- Utenti gestiti IAM tramite un provider di identità:

  Crea un ruolo per la federazione delle identità. Segui le istruzioni riportate in Creazione di un ruolo per un provider di identità di terze parti (federazione) nella Guida per l'IAMutente.
- IAMutenti:
  - Crea un ruolo che l'utente possa assumere. Segui le istruzioni riportate nella sezione Creazione di un ruolo per un IAM utente nella Guida per l'IAMutente.
  - (Non consigliato) Collega una policy direttamente a un utente o aggiungi un utente a un gruppo di utenti. Segui le istruzioni riportate in Aggiungere autorizzazioni a un utente (console) nella Guida per l'IAMutente.

Per allegare un documento IAM di policy a un gruppo IAM

1. Seleziona Groups (Gruppi) nel riquadro di navigazione principale.
2. Seleziona la scheda Permissions (Autorizzazioni) nel gruppo scelto.
3. Scegli Collega policy.
4. Seleziona il documento di policy creato precedentemente, quindi seleziona Attach policy (Collega policy).

APIAffinché Gateway possa chiamare altri AWS servizi per tuo conto, crea un IAM ruolo del tipo Amazon API Gateway.

Per creare un tipo di ruolo Amazon API Gateway

1. Seleziona Roles (Ruoli) nel riquadro di navigazione principale.
2. Scegli Crea nuovo ruolo.
3. Digita un nome in Role name (Nome ruolo), quindi scegli Next Step (Fase successiva).
4. In Seleziona tipo di ruolo, in Ruoli AWS di servizio, scegli Seleziona accanto ad Amazon API Gateway.
5. Scegli una politica di IAM autorizzazioni gestite disponibile, ad esempio A mazonAPIGateway PushToCloudWatchLog se desideri che API Gateway registri le metriche CloudWatch, sotto Allach Policy, quindi scegli Next Step.
6. In Trusted Entities (Entità affidabili) verifica che apigateway.amazonaws.com sia presente come voce dell'elenco quindi scegli Create Role (Crea ruolo).
7. Nel ruolo appena creato scegli la scheda Permissions (Autorizzazioni), quindi Attach Policy (Collega policy).
8. Scegli il documento di IAM policy personalizzato creato in precedenza, quindi scegli Allega politica.

Usa le politiche VPC degli endpoint per uso privato APIs in API Gateway

Per migliorare la sicurezza dei dati privatiAPI, puoi creare una policy sugli VPC endpoint. Una policy per gli VPC endpoint è una politica IAM delle risorse che si allega a un VPC endpoint. Per ulteriori informazioni, vedere Controllo dell'accesso ai servizi con VPC gli endpoint.

Potresti voler creare una policy per gli VPC endpoint per eseguire le seguenti attività.

- Consenti solo a determinate organizzazioni o risorse di accedere al tuo VPC endpoint e richiamare il tuo. API
- Utilizza un'unica policy ed evita politiche basate sulla sessione o sui ruoli per controllare il traffico verso il tuo. API
- Rafforza il perimetro di sicurezza della tua applicazione durante la migrazione da locale a. AWS

VPCconsiderazioni sulla politica degli endpoint

Di seguito sono riportate alcune considerazioni relative alla politica degli endpointVPC.

- L'identità dell'invoker viene valutata in base al valore dell'intestazione `Authorization`. A seconda del `authorizationType` in uso, è possibile che si verifichi un errore `403 IncompleteSignatureException` o `403 InvalidSignatureException`. La tabella seguente mostra i valori dell'intestazione `Authorization` per ogni `authorizationType`.

`authorizationType`	**`Authorization`** intestazione valutata?	Valori di intestazione consentiti **`Authorization`**
`NONE` con la policy predefinita di accesso completo	No	Non passato
`NONE` con una policy di accesso personalizzata	Sì	Deve essere un valore SigV4 valido
`IAM`	Sì	Deve essere un valore SigV4 valido
`CUSTOM` o `COGNITO_USER_POOLS`	No	Non passato

- Se una politica limita l'accesso a un IAM principale specifico, ad esempio`arn:aws:iam::account-id:role/developer`, devi impostare il `authorizationType` tuo API metodo su o. `AWS_IAM` `NONE` Per ulteriori istruzioni su come impostare il metodo `authorizationType` for a, consultathe section called “Metodi”.
- VPCle politiche degli endpoint possono essere utilizzate insieme alle politiche delle risorse di API Gateway. La politica delle risorse di API Gateway specifica quali principali possono accedere a. API La policy dell'endpoint specifica chi può accedere VPC e quali APIs possono essere richiamati dall'endpoint. VPC L'utente privato API necessita di una policy in materia di risorse, ma non è necessario creare una policy personalizzata VPC per gli endpoint.

VPCesempi di policy sugli endpoint

Puoi creare policy per gli endpoint Amazon Virtual Private Cloud per Amazon API Gateway in cui puoi specificare quanto segue.

- Il principale che può eseguire azioni.
- Le operazioni che possono essere eseguite.

- Le risorse su cui è possibile eseguire le operazioni.

Per collegare la policy all'VPCendpoint, dovrai utilizzare la VPC console. Per ulteriori informazioni, consulta Controllo dell'accesso ai servizi con gli VPC endpoint.

Esempio 1: policy sugli VPC endpoint che concede l'accesso a due APIs

La seguente politica di esempio concede l'accesso solo a due dispositivi specifici APIs tramite l'VPCendpoint a cui è associata la policy.

```
{
    "Statement": [
        {
            "Principal": "*",
            "Action": [
                "execute-api:Invoke"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:execute-api:us-east-1:123412341234:a1b2c3d4e5/*",
                "arn:aws:execute-api:us-east-1:123412341234:aaaaa11111/*"
            ]
        }
    ]
}
```

Esempio 2: la policy VPC degli endpoint che concede l'accesso ai metodi GET

La seguente politica di esempio concede agli utenti l'accesso ai `GET` metodi per uno specifico dispositivo API tramite l'VPCendpoint a cui è allegata la policy.

```
{
    "Statement": [
        {
            "Principal": "*",
            "Action": [
                "execute-api:Invoke"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:execute-api:us-east-1:123412341234:a1b2c3d4e5/stageName/GET/*"
            ]
```

```
        }
    ]
}
```

Esempio 3: politica VPC degli endpoint che concede a un utente specifico l'accesso a un determinato API

La seguente politica di esempio concede a un utente specifico l'accesso a uno specifico API tramite l'VPCendpoint a cui è associata la policy.

In questo caso, poiché la policy limita l'accesso a IAM principali specifici, è necessario impostare il `authorizationType` metodo su o. `AWS_IAM` `NONE`

```
{
    "Statement": [
        {
            "Principal": {
                "AWS": [
                    "arn:aws:iam::123412341234:user/MyUser"
                ]
            },
            "Action": [
                "execute-api:Invoke"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:execute-api:us-east-1:123412341234:a1b2c3d4e5/*"
            ]
        }
    ]
}
```

Usa i tag per controllare l'accesso alle API REST in API Gateway

L'autorizzazione per accedere alle API REST può essere ottimizzata usando il controllo degli accessi basato sugli attributi nelle policy IAM.

Per ulteriori informazioni, consulta the section called “Controllo dell'accesso basato sugli attributi”.

Usa gli API autorizzatori Gateway Lambda

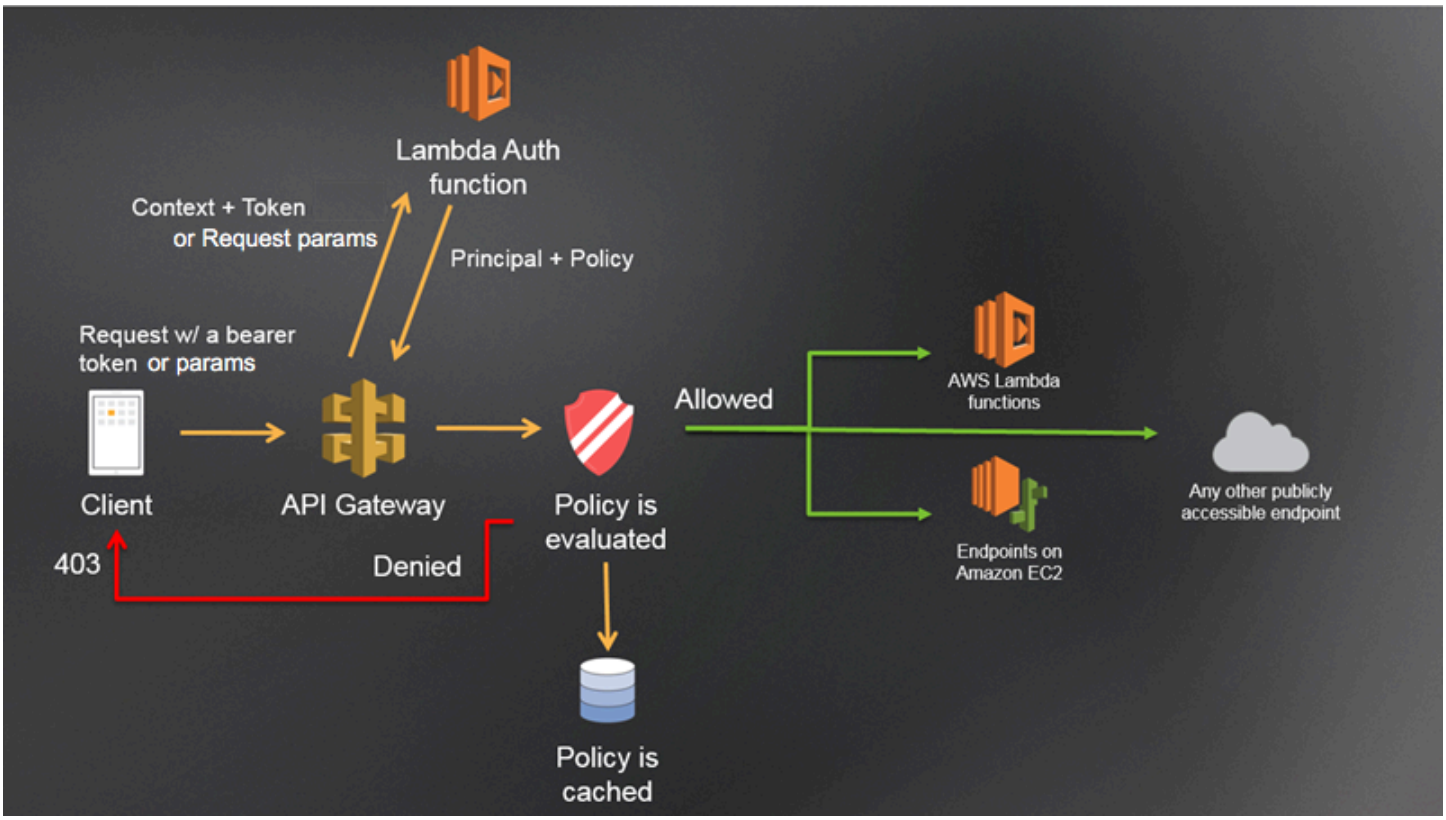
Usa un autorizzatore Lambda (precedentemente noto come autorizzatore personalizzato) per controllare l'accesso al tuo. API Quando un client richiede il tuo API metodo, API Gateway chiama

l'autorizzatore Lambda. L'autorizzatore Lambda accetta l'identità del chiamante come input e restituisce una IAM policy come output.

Usa un autorizzatore Lambda per implementare uno schema di autorizzazione personalizzato. Lo schema può utilizzare i parametri di richiesta per determinare l'identità del chiamante o utilizzare una strategia di autenticazione con token portatore come o. OAuth SAML Crea un autorizzatore Lambda nella REST API console API Gateway, utilizzando o un. AWS CLI AWS SDK

Flusso di lavoro di autorizzazione Lambda Authorizer

Il diagramma seguente mostra il flusso di lavoro di autorizzazione per un autorizzatore Lambda.



APIFlusso di lavoro di autorizzazione Gateway Lambda

1. Il client chiama un metodo su un API GatewayAPI, passando un token bearer o richiedendo parametri.
2. APIIl gateway verifica se la richiesta del metodo è configurata con un autorizzatore Lambda. In tal caso, API Gateway chiama la funzione Lambda.
3. La funzione Lambda autentica il chiamante. La funzione può autenticarsi nei seguenti modi:
   - Chiamando un OAuth provider per ottenere un token di OAuth accesso.

- Chiamando un SAML provider per ottenere un'SAMLaffermazione.
- Generando una IAM politica basata sui valori dei parametri di richiesta.
- Recuperando le credenziali da un database.

4. La funzione Lambda restituisce una IAM policy e un identificatore principale. Se la funzione Lambda non restituisce tali informazioni, la chiamata ha esito negativo.
5. APIGateway valuta la politica. IAM

   - Se l'accesso viene negato, API Gateway restituisce un codice di HTTP stato adeguato, ad esempio`403 ACCESS_DENIED`.
   - Se l'accesso è consentito, API Gateway richiama il metodo.

     Se abiliti la memorizzazione nella cache delle autorizzazioni, API Gateway memorizza nella cache la policy in modo che la funzione di autorizzazione Lambda non venga richiamata nuovamente.

È possibile personalizzare le risposte o il gateway`403 ACCESS_DENIED`. `401 UNAUTHORIZED` Per ulteriori informazioni, consulta the section called “Risposte del gateway”.

Scelta di un tipo di autorizzatore Lambda

Esistono due tipi di autorizzazioni Lambda:

Richiedi l'autorizzazione Lambda basata su parametri (authorizer) **`REQUEST`**

Un `REQUEST` autorizzatore riceve l'identità del chiamante in una combinazione di intestazioni, parametri della stringa di query e variabili. `stageVariables$context` È possibile utilizzare un programma di `REQUEST` autorizzazione per creare politiche dettagliate basate sulle informazioni provenienti da più fonti di identità, come le variabili e di contesto. `$context.path` `$context.httpMethod`

Se si attiva la memorizzazione nella cache delle autorizzazioni per un `REQUEST` autorizzatore, API Gateway verifica che tutte le fonti di identità specificate siano presenti nella richiesta. Se una fonte di identificazione specificata è mancante, nulla o vuota, API Gateway restituisce una `401 Unauthorized` HTTP risposta senza chiamare la funzione di autorizzazione Lambda. Quando vengono definite più fonti di identità, vengono tutte utilizzate per derivare la chiave della cache dell'autorizzatore, mantenendo l'ordine. È possibile definire una chiave di cache dettagliata utilizzando più fonti di identità.

Se modificate una qualsiasi delle parti chiave della cache e ridistribuite la vostraAPI, l'autorizzatore scarta il documento di policy memorizzato nella cache e ne genera uno nuovo.

Se disattivi la memorizzazione nella cache delle autorizzazioni per un `REQUEST` autorizzatore, API Gateway passa direttamente la richiesta alla funzione Lambda.

Autorizzatore Lambda basato su token (autorizzatore) **`TOKEN`**

Un `TOKEN` autorizzatore riceve l'identità del chiamante in un token portante, ad esempio un Web Token () o un JSON token. JWT OAuth

Se si attiva la memorizzazione nella cache delle autorizzazioni per un `TOKEN` autorizzatore, il nome dell'intestazione specificato nella fonte del token diventa la chiave della cache.

Inoltre, puoi utilizzare la convalida dei token per inserire una dichiarazione. RegEx APIGateway esegue la convalida iniziale del token di input rispetto a questa espressione e richiama la funzione di autorizzazione Lambda in caso di convalida riuscita. Questo aiuta a ridurre le chiamate al tuo. API

La `IdentityValidationExpression` proprietà è supportata solo per `TOKEN` gli autorizzatori. Per ulteriori informazioni, consulta the section called “`x-amazon-apigateway-authorizer`”.

> **Note**
>
> Ti consigliamo di utilizzare un `REQUEST` autorizzatore per controllare l'accesso al tuo. API È possibile controllare l'accesso alla propria identità in API base a più fonti di identità quando si utilizza un `REQUEST` autorizzatore, rispetto a un'unica fonte di identità quando si utilizza un `TOKEN` autorizzatore. Inoltre, puoi separare le chiavi della cache utilizzando più fonti di identità per un `REQUEST` autorizzatore.

Esempio di funzione Lambda **`REQUEST`** dell'autorizzatore

Il codice di esempio seguente crea una funzione di autorizzazione Lambda che consente una richiesta se l'`HeaderAuth1`intestazione, il parametro di `QueryString1` query e la variabile stage forniti dal client corrispondono `StageVar1` tutti ai valori specificati di, e, rispettivamente. `headerValue1` `queryValue1` `stageValue1`

Node.js

```
// A simple request-based authorizer example to demonstrate how to use request
// parameters to allow or deny a request. In this example, a request is
// authorized if the client-supplied HeaderAuth1 header, QueryString1
// query parameter, and stage variable of StageVar1 all match
// specified values of 'headerValue1', 'queryValue1', and 'stageValue1',
// respectively.

export const handler = function(event, context, callback) {
    console.log('Received event:', JSON.stringify(event, null, 2));

    // Retrieve request parameters from the Lambda function input:
    var headers = event.headers;
    var queryStringParameters = event.queryStringParameters;
    var pathParameters = event.pathParameters;
    var stageVariables = event.stageVariables;

    // Parse the input for the parameter values
    var tmp = event.methodArn.split(':');
    var apiGatewayArnTmp = tmp[5].split('/');
    var awsAccountId = tmp[4];
    var region = tmp[3];
    var restApiId = apiGatewayArnTmp[0];
    var stage = apiGatewayArnTmp[1];
    var method = apiGatewayArnTmp[2];
    var resource = '/'; // root resource
    if (apiGatewayArnTmp[3]) {
        resource += apiGatewayArnTmp[3];
    }

    // Perform authorization to return the Allow policy for correct parameters and
    // the 'Unauthorized' error, otherwise.


    if (headers.HeaderAuth1 === "headerValue1"
        && queryStringParameters.QueryString1 === "queryValue1"
        && stageVariables.StageVar1 === "stageValue1") {
        callback(null, generateAllow('me', event.methodArn));
    }  else {
        callback("Unauthorized");
    }
}
```

```
// Help function to generate an IAM policy
var generatePolicy = function(principalId, effect, resource) {
    // Required output:
    var authResponse = {};
    authResponse.principalId = principalId;
    if (effect && resource) {
        var policyDocument = {};
        policyDocument.Version = '2012-10-17'; // default version
        policyDocument.Statement = [];
        var statementOne = {};
        statementOne.Action = 'execute-api:Invoke'; // default action
        statementOne.Effect = effect;
        statementOne.Resource = resource;
        policyDocument.Statement[0] = statementOne;
        authResponse.policyDocument = policyDocument;
    }
    // Optional output with custom properties of the String, Number or Boolean type.
    authResponse.context = {
        "stringKey": "stringval",
        "numberKey": 123,
        "booleanKey": true
    };
    return authResponse;
}

var generateAllow = function(principalId, resource) {
    return generatePolicy(principalId, 'Allow', resource);
}

var generateDeny = function(principalId, resource) {
    return generatePolicy(principalId, 'Deny', resource);
}
```

Python

```
# A simple request-based authorizer example to demonstrate how to use request
# parameters to allow or deny a request. In this example, a request is
# authorized if the client-supplied headerauth1 header, QueryString1
# query parameter, and stage variable of StageVar1 all match
# specified values of 'headerValue1', 'queryValue1', and 'stageValue1',
# respectively.

import json
```

```
def lambda_handler(event, context):
    print(event)

    # Retrieve request parameters from the Lambda function input:
    headers = event['headers']
    queryStringParameters = event['queryStringParameters']
    pathParameters = event['pathParameters']
    stageVariables = event['stageVariables']

    # Parse the input for the parameter values
    tmp = event['methodArn'].split(':')
    apiGatewayArnTmp = tmp[5].split('/')
    awsAccountId = tmp[4]
    region = tmp[3]
    restApiId = apiGatewayArnTmp[0]
    stage = apiGatewayArnTmp[1]
    method = apiGatewayArnTmp[2]
    resource = '/'

    if (apiGatewayArnTmp[3]):
        resource += apiGatewayArnTmp[3]

    # Perform authorization to return the Allow policy for correct parameters
    # and the 'Unauthorized' error, otherwise.

    if (headers['HeaderAuth1'] == "headerValue1" and
 queryStringParameters['QueryString1'] == "queryValue1" and
 stageVariables['StageVar1'] == "stageValue1"):
        response = generateAllow('me', event['methodArn'])
        print('authorized')
        return response
    else:
        print('unauthorized')
        raise Exception('Unauthorized') # Return a 401 Unauthorized response

    # Help function to generate IAM policy


def generatePolicy(principalId, effect, resource):
    authResponse = {}
    authResponse['principalId'] = principalId
    if (effect and resource):
```

```
        policyDocument = {}
        policyDocument['Version'] = '2012-10-17'
        policyDocument['Statement'] = []
        statementOne = {}
        statementOne['Action'] = 'execute-api:Invoke'
        statementOne['Effect'] = effect
        statementOne['Resource'] = resource
        policyDocument['Statement'] = [statementOne]
        authResponse['policyDocument'] = policyDocument

    authResponse['context'] = {
        "stringKey": "stringval",
        "numberKey": 123,
        "booleanKey": True
    }

    return authResponse


def generateAllow(principalId, resource):
    return generatePolicy(principalId, 'Allow', resource)


def generateDeny(principalId, resource):
    return generatePolicy(principalId, 'Deny', resource)
```

In questo esempio, la funzione di autorizzazione Lambda controlla i parametri di input e si comporta nel modo seguente:

- Se tutti i valori dei parametri richiesti corrispondono ai valori previsti, la funzione di autorizzazione restituisce una `200 OK` HTTP risposta e una IAM politica simili alle seguenti e la richiesta del metodo ha esito positivo:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "execute-api:Invoke",
      "Effect": "Allow",
      "Resource": "arn:aws:execute-api:us-east-1:123456789012:ivdtdhp7b5/
ESTestInvoke-stage/GET/"
    }
```

```
  ]
}
```

- In caso contrario, la funzione di autorizzazione restituisce una `401 Unauthorized` HTTP risposta e la richiesta del metodo ha esito negativo.

Oltre a restituire una IAM policy, la funzione di autorizzazione Lambda deve restituire anche l'identificatore principale del chiamante. Facoltativamente, può restituire un `context` oggetto contenente informazioni aggiuntive che possono essere passate al backend di integrazione. Per ulteriori informazioni, consulta Output da un autorizzatore API Gateway Lambda.

Nel codice di produzione, potrebbe essere necessario autenticare l'utente prima di concedere l'autorizzazione. È possibile aggiungere la logica di autenticazione nella funzione Lambda chiamando un provider di autenticazione come indicato nella documentazione relativa a tale provider.

Esempio di funzione Lambda **`TOKEN`** dell'autorizzatore

Il codice di esempio seguente crea una funzione di autorizzazione `TOKEN` Lambda che consente a un chiamante di richiamare un metodo se il valore del token fornito dal client è. `allow` Al chiamante non è consentito richiamare la richiesta se il valore del token è. `deny` Se il valore del token è `unauthorized` o è una stringa vuota, la funzione di autorizzazione restituisce una risposta. `401 UNAUTHORIZED`

Node.js

```
// A simple token-based authorizer example to demonstrate how to use an
 authorization token
// to allow or deny a request. In this example, the caller named 'user' is allowed
 to invoke
// a request if the client-supplied token value is 'allow'. The caller is not
 allowed to invoke
// the request if the token value is 'deny'. If the token value is 'unauthorized' or
 an empty
// string, the authorizer function returns an HTTP 401 status code. For any other
 token value,
// the authorizer returns an HTTP 500 status code.
// Note that token values are case-sensitive.

export const handler =  function(event, context, callback) {
    var token = event.authorizationToken;
    switch (token) {
```

```
        case 'allow':
            callback(null, generatePolicy('user', 'Allow', event.methodArn));
            break;
        case 'deny':
            callback(null, generatePolicy('user', 'Deny', event.methodArn));
            break;
        case 'unauthorized':
            callback("Unauthorized");   // Return a 401 Unauthorized response
            break;
        default:
            callback("Error: Invalid token"); // Return a 500 Invalid token response
    }
};

// Help function to generate an IAM policy
var generatePolicy = function(principalId, effect, resource) {
    var authResponse = {};

    authResponse.principalId = principalId;
    if (effect && resource) {
        var policyDocument = {};
        policyDocument.Version = '2012-10-17';
        policyDocument.Statement = [];
        var statementOne = {};
        statementOne.Action = 'execute-api:Invoke';
        statementOne.Effect = effect;
        statementOne.Resource = resource;
        policyDocument.Statement[0] = statementOne;
        authResponse.policyDocument = policyDocument;
    }

    // Optional output with custom properties of the String, Number or Boolean type.
    authResponse.context = {
        "stringKey": "stringval",
        "numberKey": 123,
        "booleanKey": true
    };
    return authResponse;
}
```

Python

```
# A simple token-based authorizer example to demonstrate how to use an authorization
 token
# to allow or deny a request. In this example, the caller named 'user' is allowed to
 invoke
# a request if the client-supplied token value is 'allow'. The caller is not allowed
 to invoke
# the request if the token value is 'deny'. If the token value is 'unauthorized' or
 an empty
# string, the authorizer function returns an HTTP 401 status code. For any other
 token value,
# the authorizer returns an HTTP 500 status code.
# Note that token values are case-sensitive.

import json


def lambda_handler(event, context):
    token = event['authorizationToken']
    if token == 'allow':
        print('authorized')
        response = generatePolicy('user', 'Allow', event['methodArn'])
    elif token == 'deny':
        print('unauthorized')
        response = generatePolicy('user', 'Deny', event['methodArn'])
    elif token == 'unauthorized':
        print('unauthorized')
        raise Exception('Unauthorized')  # Return a 401 Unauthorized response
        return 'unauthorized'
    try:
        return json.loads(response)
    except BaseException:
        print('unauthorized')
        return 'unauthorized'  # Return a 500 error


def generatePolicy(principalId, effect, resource):
    authResponse = {}
    authResponse['principalId'] = principalId
    if (effect and resource):
        policyDocument = {}
        policyDocument['Version'] = '2012-10-17'
        policyDocument['Statement'] = []
```

```
        statementOne = {}
        statementOne['Action'] = 'execute-api:Invoke'
        statementOne['Effect'] = effect
        statementOne['Resource'] = resource
        policyDocument['Statement'] = [statementOne]
        authResponse['policyDocument'] = policyDocument
    authResponse['context'] = {
        "stringKey": "stringval",
        "numberKey": 123,
        "booleanKey": True
    }
    authResponse_JSON = json.dumps(authResponse)
    return authResponse_JSON
```

In questo esempio, quando API riceve una richiesta di metodo, API Gateway passa il token di origine a questa funzione di autorizzazione Lambda nell'attributo. `event.authorizationToken` La funzione di autorizzazione Lambda legge il token e si comporta nel modo seguente:

- Se il valore del token è`allow`, la funzione di autorizzazione restituisce una `200 OK` HTTP risposta e una IAM politica simili alle seguenti e la richiesta del metodo ha esito positivo:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "execute-api:Invoke",
      "Effect": "Allow",
      "Resource": "arn:aws:execute-api:us-east-1:123456789012:ivdtdhp7b5/
ESTestInvoke-stage/GET/"
    }
  ]
}
```

- Se il valore del token è`deny`, la funzione di autorizzazione restituisce una `200 OK` HTTP risposta e una `Deny` IAM politica simili alle seguenti e la richiesta del metodo ha esito negativo:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "execute-api:Invoke",
```

```
      "Effect": "Deny",
      "Resource": "arn:aws:execute-api:us-east-1:123456789012:ivdtdhp7b5/
ESTestInvoke-stage/GET/"
    }
  ]
}
```

> **Note**
>
> Al di fuori dell'ambiente di test, API Gateway restituisce una `403 Forbidden` HTTP risposta e la richiesta del metodo ha esito negativo.

- Se il valore del token è `unauthorized` o è una stringa vuota, la funzione di autorizzazione restituisce una `401 Unauthorized` HTTP risposta e la chiamata al metodo ha esito negativo.
- Se il token ha un valore diverso dai precedenti, il client riceve una risposta `500 Invalid token` e la chiamata al metodo ha esito negativo.

Oltre a restituire una IAM policy, la funzione di autorizzazione Lambda deve restituire anche l'identificatore principale del chiamante. Facoltativamente, può restituire un `context` oggetto contenente informazioni aggiuntive che possono essere passate al backend di integrazione. Per ulteriori informazioni, consulta Output da un autorizzatore API Gateway Lambda.

Nel codice di produzione, potrebbe essere necessario autenticare l'utente prima di concedere l'autorizzazione. È possibile aggiungere la logica di autenticazione nella funzione Lambda chiamando un provider di autenticazione come indicato nella documentazione relativa a tale provider.

Esempi aggiuntivi di funzioni di autorizzazione Lambda

L'elenco seguente mostra altri esempi di funzioni di autorizzazione Lambda. Puoi creare una funzione Lambda nello stesso account, o in un account diverso, da dove hai creato il tuo. API

Per l'esempio precedente di funzioni Lambda, puoi utilizzare le funzioni integrate AWSLambdaBasicExecutionRole, poiché queste funzioni non chiamano altri AWS servizi. Se la tua funzione Lambda chiama altri AWS servizi, dovrai assegnare un ruolo di IAM esecuzione alla funzione Lambda. Per creare il ruolo, segui le istruzioni in Ruolo di esecuzione AWS Lambda.

Esempi aggiuntivi di funzioni di autorizzazione Lambda

- Per un esempio di applicazione, vedete Open Banking Brazil - Authorization Samples on. GitHub

- Per altri esempi di funzioni Lambda, vedi aws-apigateway-lambda-authorizer-blueprints on. GitHub
- Puoi creare un sistema di autorizzazione Lambda che autentichi gli utenti utilizzando i pool di utenti di Amazon Cognito e autorizzi i chiamanti in base a un policy store utilizzando Autorizzazioni verificate. Per ulteriori informazioni, consulta the section called “Controlla l'accesso in base agli attributi di un'identità con autorizzazioni verificate”.
- La console Lambda fornisce un blueprint Python, che puoi usare scegliendo Usa un blueprint e scegliendo il blueprint. api-gateway-authorizer-python

Configurare un autorizzatore API Gateway Lambda

Dopo aver creato una funzione Lambda, configuri la funzione Lambda come autorizzatore per la tua. API Quindi configuri il metodo per richiamare l'autorizzatore Lambda per determinare se un chiamante può richiamare il tuo metodo. Puoi creare una funzione Lambda nello stesso account, o in un account diverso, da dove hai creato il tuo. API

Puoi testare l'autorizzatore Lambda utilizzando gli strumenti integrati nella console API Gateway o utilizzando Postman. Per istruzioni su come usare Postman per testare la funzione di autorizzazione Lambda, consulta. the section called “Chiama un utente API con autorizzatori Lambda”

Configurare un autorizzatore Lambda (console)

La procedura seguente mostra come creare un autorizzatore Lambda nella console Gateway. API REST API Per ulteriori informazioni sui diversi tipi di autorizzatori Lambda, consulta. the section called “Scelta di un tipo di autorizzatore Lambda”

REQUEST authorizer

Per configurare un autorizzatore **`REQUEST`** Lambda

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Seleziona unAPI, quindi scegli Autorizzatori.
3. Scegli Create Authorizer (Crea autorizzazioni).
4. In Nome del provider di autorizzazioni, immetti il nome di un provider di autorizzazioni.
5. In Tipo di autorizzazione, seleziona Lambda.
6. Per la funzione Lambda, seleziona il Regione AWS luogo in cui hai creato la funzione di autorizzazione Lambda, quindi inserisci il nome della funzione.

7. Lascia vuoto il ruolo di richiamo Lambda per consentire alla REST API console API Gateway di impostare una policy basata sulle risorse. La policy concede a API Gateway le autorizzazioni per richiamare la funzione di autorizzazione Lambda. Puoi anche scegliere di inserire il nome di un IAM ruolo per consentire a API Gateway di richiamare la funzione di autorizzazione Lambda. Per un ruolo di esempio, vedi. Creazione di un ruolo IAM assumibile
8. In Payload evento Lambda, scegli Richiesta.
9. In Tipo di origine identità, seleziona un tipo di parametro. I tipi di parametri supportati sono `Header`, `Query string`, `Stage variable` e `Context`. Per aggiungere altre origini di identità, scegli Aggiungi parametro.
10. Per memorizzare nella cache la policy di autorizzazione generata dalla funzione di autorizzazione, attiva l'opzione Autorizzazione caching. Quando la memorizzazione nella cache delle policy è abilitata, è possibile modificare il TTLvalore. L'impostazione TTLsu zero disabilita la memorizzazione nella cache delle politiche.

    Se abiliti la memorizzazione nella cache, l'autorizzatore deve restituire una politica applicabile a tutti i metodi di un. API Per applicare una politica specifica per il metodo, utilizza le variabili di contesto e. `$context.path` `$context.httpMethod`
11. Scegli Create Authorizer (Crea autorizzazioni).

TOKEN authorizer

Per configurare un autorizzatore **`TOKEN`** Lambda

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Seleziona unAPI, quindi scegli Autorizzatori.
3. Scegli Create Authorizer (Crea autorizzazioni).
4. In Nome del provider di autorizzazioni, immetti il nome di un provider di autorizzazioni.
5. In Tipo di autorizzazione, seleziona Lambda.
6. Per la funzione Lambda, seleziona il Regione AWS luogo in cui hai creato la funzione di autorizzazione Lambda, quindi inserisci il nome della funzione.
7. Lascia vuoto il ruolo di richiamo Lambda per consentire alla REST API console API Gateway di impostare una policy basata sulle risorse. La policy concede a API Gateway le autorizzazioni per richiamare la funzione di autorizzazione Lambda. Puoi anche scegliere di inserire il nome di un IAM ruolo per consentire a API Gateway di richiamare la funzione di autorizzazione Lambda. Per un ruolo di esempio, vedi. Creazione di un ruolo IAM assumibile

8. In Payload evento Lambda, scegli Token.
9. In Origine token, immetti il nome dell'intestazione contenente il token di autorizzazione. Il chiamante deve includere un'intestazione con questo nome per inviare il token di autorizzazione all'autorizzatore Lambda.
10. (Facoltativo) Per la convalida del token, inserisci una dichiarazione. RegEx APIGateway esegue la convalida iniziale del token di input rispetto a questa espressione e richiama l'autorizzatore dopo la convalida riuscita.
11. Per memorizzare nella cache la policy di autorizzazione generata dalla funzione di autorizzazione, attiva l'opzione Autorizzazione caching. Quando la memorizzazione della policy nella cache è abilitata, il nome dell'intestazione specificato in Origine token diventa la chiave della cache. Quando la memorizzazione nella cache delle policy è abilitata, è possibile modificare il valore. TTL L'impostazione TTLsu zero disabilita la memorizzazione nella cache delle politiche.

    Se abiliti la memorizzazione nella cache, l'autorizzatore deve restituire una politica applicabile a tutti i metodi di un. API Per applicare una politica specifica del metodo, puoi disattivare la memorizzazione nella cache delle autorizzazioni.
12. Scegli Create Authorizer (Crea autorizzazioni).

Dopo aver creato l'autorizzatore Lambda, puoi testarlo. La procedura seguente mostra come testare l'autorizzatore Lambda.

REQUEST authorizer

Per testare un autorizzatore **`REQUEST`** Lambda

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Seleziona il nome del tuo autorizzatore.
3. In Test authorizer, inserisci un valore per la tua fonte di identità.

   Se utilizzi ilthe section called “Esempio di funzione Lambda `REQUEST` dell'autorizzatore”, procedi come segue:

   a. Seleziona Intestazione e immetti **`headerValue1`**, quindi scegli Aggiungi parametro.
   b. In Tipo di origine identità, seleziona Stringa di query e immetti **`queryValue1`**, quindi scegli Aggiungi parametro.

c. In Tipo di origine identità, seleziona Variabile di fase e immetti **`stageValue1`**.

Non puoi modificare le variabili di contesto per la chiamata di test, ma puoi modificare il modello di evento di test APIGateway Authorizer per la tua funzione Lambda. Quindi, puoi testare la tua funzione di autorizzazione Lambda con variabili di contesto modificate. Per ulteriori informazioni, consulta Testare le funzioni Lambda nella console nella Guida per gli AWS Lambda sviluppatori.

4. Scegli Testa autorizzazioni.

TOKEN authorizer

Per testare un autorizzatore **`TOKEN`** Lambda

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Seleziona il nome del tuo autorizzatore.
3. In Test authorizer, inserisci un valore per il tuo token.

   Se utilizzi ilthe section called “Esempio di funzione Lambda `TOKEN` dell'autorizzatore”, procedi come segue:

   - Per il authorizationToken, inserisci**`allow`**.
4. Scegli Testa autorizzazioni.

   Se l'autorizzatore Lambda nega correttamente una richiesta nell'ambiente di test, il test risponde con una risposta. `200 OK` HTTP Tuttavia, al di fuori dell'ambiente di test, API Gateway restituisce una `403 Forbidden` HTTP risposta e la richiesta del metodo fallisce.

Configurare un autorizzatore Lambda ()AWS CLI

Il seguente comando create-authorizer mostra come creare un autorizzatore Lambda utilizzando. AWS CLI

REQUEST authorizer

L'esempio seguente crea un `REQUEST` authorizer e utilizza l'`Authorizer`header e la variabile di contesto come fonti di identità: `accountId`

```
aws apigateway create-authorizer \
```

```
    --rest-api-id 1234123412 \
    --name 'First_Request_Custom_Authorizer' \
    --type REQUEST \
    --authorizer-uri 'arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123412341234:function:customAuthFunction/invocations' \
    --identity-source 'method.request.header.Authorization,context.accountId' \
    --authorizer-result-ttl-in-seconds 300
```

TOKEN authorizer

L'esempio seguente crea un `TOKEN` autorizzatore e utilizza l'`Authorization`intestazione come fonte di identità:

```
aws apigateway create-authorizer \
    --rest-api-id 1234123412 \
    --name 'First_Token_Custom_Authorizer' \
    --type TOKEN \
    --authorizer-uri 'arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123412341234:function:customAuthFunction/invocations' \
    --identity-source 'method.request.header.Authorization' \
    --authorizer-result-ttl-in-seconds 300
```

Dopo aver creato l'autorizzatore Lambda, puoi testarlo. Il test-invoke-authorizercomando seguente mostra come testare l'autorizzatore Lambda:

```
aws apigateway test-invoke-authorizer --rest-api-id 1234123412 \
   --authorizer-id efg1234 \
   --headers Authorization='Value'
```

Configurare un metodo per utilizzare un autorizzatore Lambda (console)

Dopo aver configurato l'autorizzatore Lambda, devi collegarlo a un metodo per il tuo. API

Per configurare un API metodo per utilizzare un autorizzatore Lambda

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Seleziona unAPI.
3. Scegli Risorse, quindi scegli un nuovo metodo o scegli un metodo esistente.
4. Nella scheda Richiesta metodo, in Impostazioni richiesta metodo, scegli Modifica.

5. In Autorizzazioni, nel menu a discesa, seleziona la funzione di autorizzazione Lambda appena creata.
6. (Facoltativo) Se desideri passare il token di autorizzazione al backend, scegli le intestazioni della HTTP richiesta. Scegli Aggiungi intestazione, quindi aggiungi il nome dell'intestazione di autorizzazione. Per Nome, inserisci il nome dell'intestazione che corrisponde al nome di origine del token che hai specificato quando hai creato l'autorizzatore Lambda per. API Questo passaggio non si applica alle autorizzazioni `REQUEST`.
7. Selezionare Salva.
8. Scegli Deploy API per distribuirlo su una fase. API Per una funzione di autorizzazione basata su `REQUEST` che usa variabili di fase, devi anche definire le variabili di fase richieste e specificare i relativi valori in Fasi.

Configurare un metodo per utilizzare un autorizzatore Lambda ()AWS CLI

Dopo aver configurato l'autorizzatore Lambda, devi collegarlo a un metodo per il tuo. API È possibile creare un nuovo metodo o utilizzare un'operazione di patch per collegare un autorizzatore a un metodo esistente.

Il seguente comando put-method mostra come creare un nuovo metodo che utilizza un autorizzatore Lambda:

```
aws apigateway put-method --rest-api-id 1234123412 \
  --resource-id a1b2c3 \
  --http-method PUT \
  --authorization-type CUSTOM \
  --authorizer-id efg1234
```

Il seguente comando update-method mostra come aggiornare un metodo esistente per utilizzare un autorizzatore Lambda:

```
aws apigateway update-method \
    --rest-api-id 1234123412 \
    --resource-id a1b2c3 \
    --http-method PUT \
    --patch-operations op="replace",path="/authorizationType",value="CUSTOM"
 op="replace",path="/authorizerId",value="efg1234"
```

Ingresso a un autorizzatore API Gateway Lambda

La sezione seguente spiega il formato dell'input da API Gateway a un autorizzatore Lambda.

Formato di input **TOKEN**

Per un autorizzatore Lambda (precedentemente noto come autorizzatore personalizzato) di questo `TOKEN` tipo, devi specificare un'intestazione personalizzata come Token Source quando configuri l'autorizzatore per il tuo. API Il API client deve passare il token di autorizzazione richiesto in quell'intestazione nella richiesta in entrata. Dopo aver ricevuto la richiesta del metodo in entrata, API Gateway estrae il token dall'intestazione personalizzata. Quindi passa il token come `authorizationToken` proprietà dell'`event`oggetto della funzione Lambda, oltre al metodo ARN come proprietà: `methodArn`

```
{
    "type":"TOKEN",
    "authorizationToken":"{caller-supplied-token}",
    "methodArn":"arn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/{httpVerb}/
[{resource}/[{child-resources}]]"
}
```

In questo esempio la proprietà `type` specifica il tipo di autorizzazioni, ovvero `TOKEN`. L'oggetto *`{caller-supplied-token}`* deriva dall'intestazione di autorizzazioni in una richiesta client e può essere qualsiasi valore di stringa. `methodArn`È la richiesta ARN del metodo in entrata e viene popolata da API Gateway in base alla configurazione dell'autorizzatore Lambda.

Formato di input **REQUEST**

Per un autorizzatore Lambda del `REQUEST` tipo, API Gateway passa i parametri di richiesta alla funzione Authorizer Lambda come parte dell'oggetto. `event` I parametri della richiesta includono intestazioni, parametri di percorso, parametri della stringa di query, variabili di fase e alcune variabili di contesto della richiesta. Il API chiamante può impostare i parametri del percorso, le intestazioni e i parametri della stringa di query. Lo API sviluppatore deve impostare le variabili di fase durante la API distribuzione e API Gateway fornisce il contesto della richiesta in fase di esecuzione.

> **Note**
>
> I parametri di percorso possono essere passati come parametri della richiesta alla funzione di autorizzazione Lambda, ma non possono essere utilizzati come origini di identità.

L'esempio seguente mostra un input a un `REQUEST` authorizer per un API metodo (`GET /request`) con un'integrazione proxy:

```
{
  "type": "REQUEST",
  "methodArn": "arn:aws:execute-api:us-east-1:123456789012:abcdef123/test/GET/request",
  "resource": "/request",
  "path": "/request",
  "httpMethod": "GET",
  "headers": {
    "X-AMZ-Date": "20170718T062915Z",
    "Accept": "*/*",
    "HeaderAuth1": "headerValue1",
    "CloudFront-Viewer-Country": "US",
    "CloudFront-Forwarded-Proto": "https",
    "CloudFront-Is-Tablet-Viewer": "false",
    "CloudFront-Is-Mobile-Viewer": "false",
    "User-Agent": "..."
  },
  "queryStringParameters": {
    "QueryString1": "queryValue1"
  },
  "pathParameters": {},
  "stageVariables": {
    "StageVar1": "stageValue1"
  },
  "requestContext": {
    "path": "/request",
    "accountId": "123456789012",
    "resourceId": "05c7jb",
    "stage": "test",
    "requestId": "...",
    "identity": {
      "apiKey": "...",
      "sourceIp": "...",
      "clientCert": {
        "clientCertPem": "CERT_CONTENT",
        "subjectDN": "www.example.com",
        "issuerDN": "Example issuer",
        "serialNumber": "a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1",
        "validity": {
          "notBefore": "May 28 12:30:02 2019 GMT",
          "notAfter": "Aug  5 09:36:04 2021 GMT"
```

```
        }
      }
    },
    "resourcePath": "/request",
    "httpMethod": "GET",
    "apiId": "abcdef123"
  }
}
```

`requestContext` è una mappa di coppie chiave/valore e corrisponde alla variabile $context. Il suo risultato dipende dal API risultato.

APIGateway potrebbe aggiungere nuove chiavi alla mappa. Per ulteriori informazioni sull'input della funzione Lambda nell'integrazione proxy Lambda, consulta Formato di input di una funzione Lambda per l'integrazione proxy.

Output da un autorizzatore API Gateway Lambda

L'output di una funzione di autorizzazione Lambda è un oggetto simile a un dizionario che deve includere l'identificatore dell'entità principale (`principalId`) e un documento di policy (`policyDocument`) contenente un elenco di dichiarazioni di policy. L'output può includere anche una mappa `context` contenente coppie chiave-valore. Se API utilizza un piano di utilizzo (`apiKeySource`impostato su`AUTHORIZER`), la funzione di autorizzazione Lambda deve restituire una delle API chiavi del piano di utilizzo come valore della proprietà. `usageIdentifierKey`

Di seguito è illustrato un esempio di output.

```
{
  "principalId": "yyyyyyyy", // The principal user identification associated with the
 token sent by the client.
  "policyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": "execute-api:Invoke",
        "Effect": "Allow|Deny",
        "Resource": "arn:aws:execute-
api:{regionId}:{accountId}:{apiId}/{stage}/{httpVerb}/[{resource}/[{child-resources}]]"
      }
    ]
  },
  "context": {
```

```
    "stringKey": "value",
    "numberKey": "1",
    "booleanKey": "true"
  },
  "usageIdentifierKey": "{api-key}"
}
```

Qui, una dichiarazione politica specifica se consentire o negare (`Effect`) al servizio di esecuzione API Gateway di invocare (`Action`) il metodo specificato (). API `Resource` Per specificare un tipo di risorsa (metodo), è possibile usare un carattere jolly (`*`). Per informazioni sull'impostazione di criteri validi per chiamare unAPI, vedere. Dichiarazione di riferimento sulle IAM politiche per l'esecuzione API in API Gateway

Per un metodo abilitato all'autorizzazioneARN, ad esempio`arn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/{httpVerb}/[{resource}/[{child-resources}]]`, la lunghezza massima è di 1600 byte. I valori dei parametri del percorso, la cui dimensione viene determinata in fase di esecuzione, possono far sì che la ARN lunghezza superi il limite. Quando ciò accade, il API client riceverà una `414 Request URI too long` risposta.

Inoltre, la ResourceARN, come mostrato nella dichiarazione politica emessa dall'autorizzatore, è attualmente limitata a 512 caratteri. Per questo motivo, non è necessario utilizzare URI un JWT token di lunghezza significativa in una richiestaURI. Puoi invece passare il JWT token in sicurezza nell'intestazione di una richiesta.

Puoi accedere al valore di `principalId` in un modello di mappatura usando la variabile `$context.authorizer.principalId`. Ciò è utile se desideri passare il valore al back-end. Per ulteriori informazioni, consulta `$context`Variabili per modelli di dati, autorizzatori, modelli di mappatura e registrazione degli accessi CloudWatch .

Puoi accedere al valore di `stringKey`, `numberKey` o `booleanKey` (ad esempio `"value"`, `"1"` o `"true"`) della mappa `context` in un modello di mappatura chiamando, rispettivamente, `$context.authorizer.stringKey`, `$context.authorizer.numberKey` o `$context.authorizer.booleanKey`. I valori restituiti sono tutti in formato stringa. Si noti che non è possibile impostare un JSON oggetto o un array come valore valido di alcuna chiave nella `context` mappa.

È possibile usare la mappa `context` per restituire le credenziali memorizzate nella cache dalle autorizzazioni al back-end, usando un modello di mappatura di richiesta di integrazione. In questo modo, il back-end può offrire un'esperienza utente migliore usando le credenziali memorizzate nella

cache per ridurre la necessità di accesso alle chiavi segrete e di apertura dei token di autorizzazione per ogni richiesta.

Per l'integrazione del proxy Lambda, API Gateway passa l'`context`oggetto da un autorizzatore Lambda direttamente alla funzione Lambda di backend come parte dell'input. `event` È possibile recuperare la coppia chiave/valore `context` nella funzione Lambda chiamando `$event.requestContext.authorizer.`*`key`*.

`{api-key}`rappresenta una API chiave nel piano di utilizzo dello API stage. Per ulteriori informazioni, consulta the section called “Piani di utilizzo”.

Di seguito è illustrato l'output dell'autorizzazione Lambda di esempio. L'output di esempio contiene una dichiarazione politica per bloccare (`Deny`) le chiamate al `GET` metodo per la `dev` fase di un API (`ymy8tbxw7b`) di un AWS account (`123456789012`).

```
{
  "principalId": "user",
  "policyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": "execute-api:Invoke",
        "Effect": "Deny",
        "Resource": "arn:aws:execute-api:us-west-2:123456789012:ymy8tbxw7b/dev/GET/"
      }
    ]
  }
}
```

Chiama un utente API con un autorizzatore API Gateway Lambda

Dopo aver configurato l'autorizzatore Lambda (precedentemente noto come autorizzatore personalizzato) e aver distribuito ilAPI, è necessario testarlo con l'autorizzatore Lambda abilitato. API A tale scopo, è necessario un REST client, ad esempio c o Postman. URL Per gli esempi seguenti usiamo Postman.

> **Note**
>
> Quando si chiama un metodo abilitato all'autorizzazione, API Gateway non registra la chiamata CloudWatch se il token richiesto per l'**`TOKEN`**autorizzatore non è impostato, è

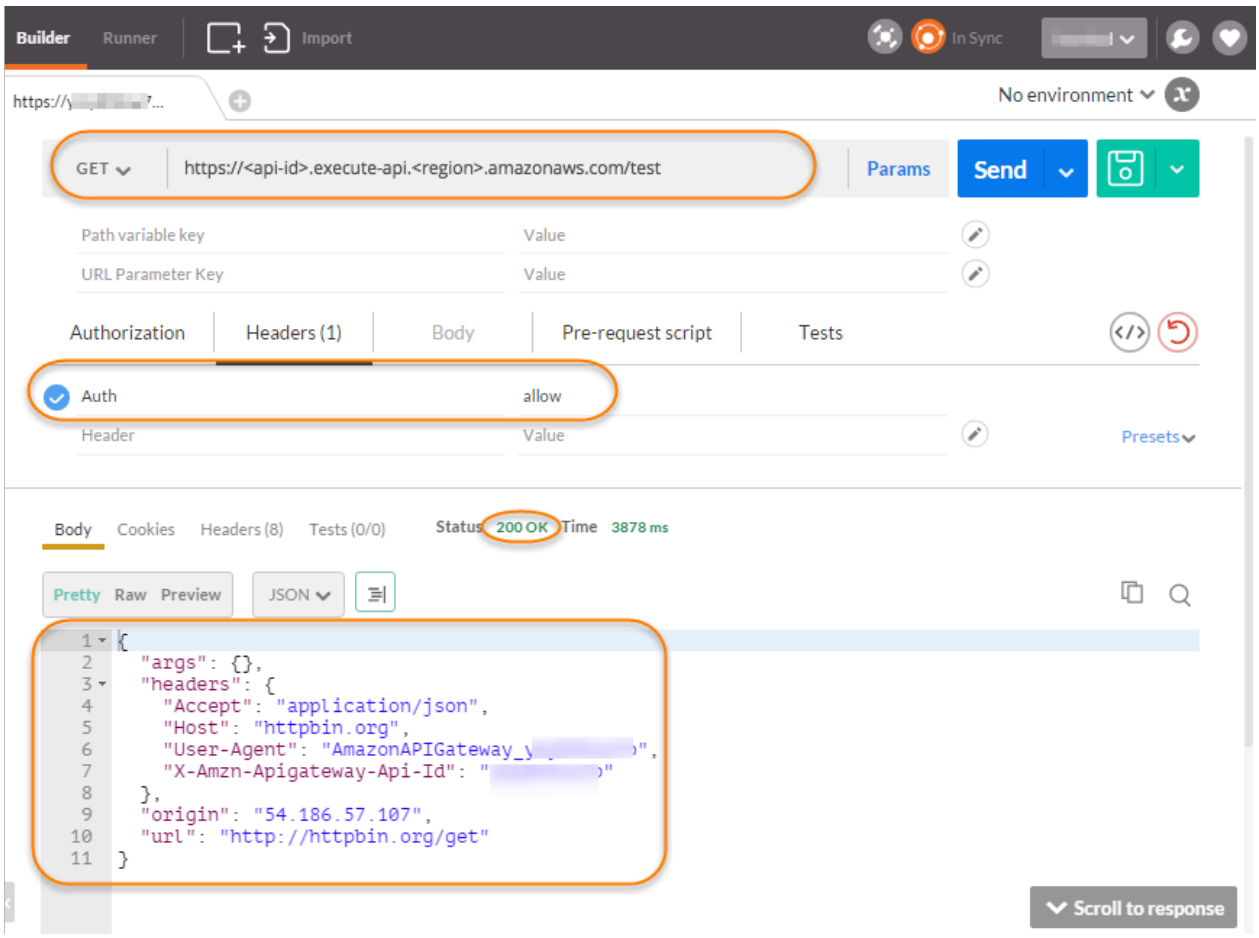
nullo o è invalidato dall'espressione di convalida Token specificata. Analogamente, API Gateway non registra la chiamata CloudWatch se una delle fonti di identità richieste per l'`REQUEST`autorizzatore non è impostata, è nulla o è vuota.

Di seguito, mostriamo come usare Postman per chiamare o testare un API con un autorizzatore `TOKEN` Lambda. Il metodo può essere applicato a una chiamata API con un `REQUEST` autorizzatore Lambda, se si specificano esplicitamente i parametri del percorso, dell'intestazione o della stringa di query richiesti.

Per chiamare un utente con l'autorizzatore personalizzato API **`TOKEN`**

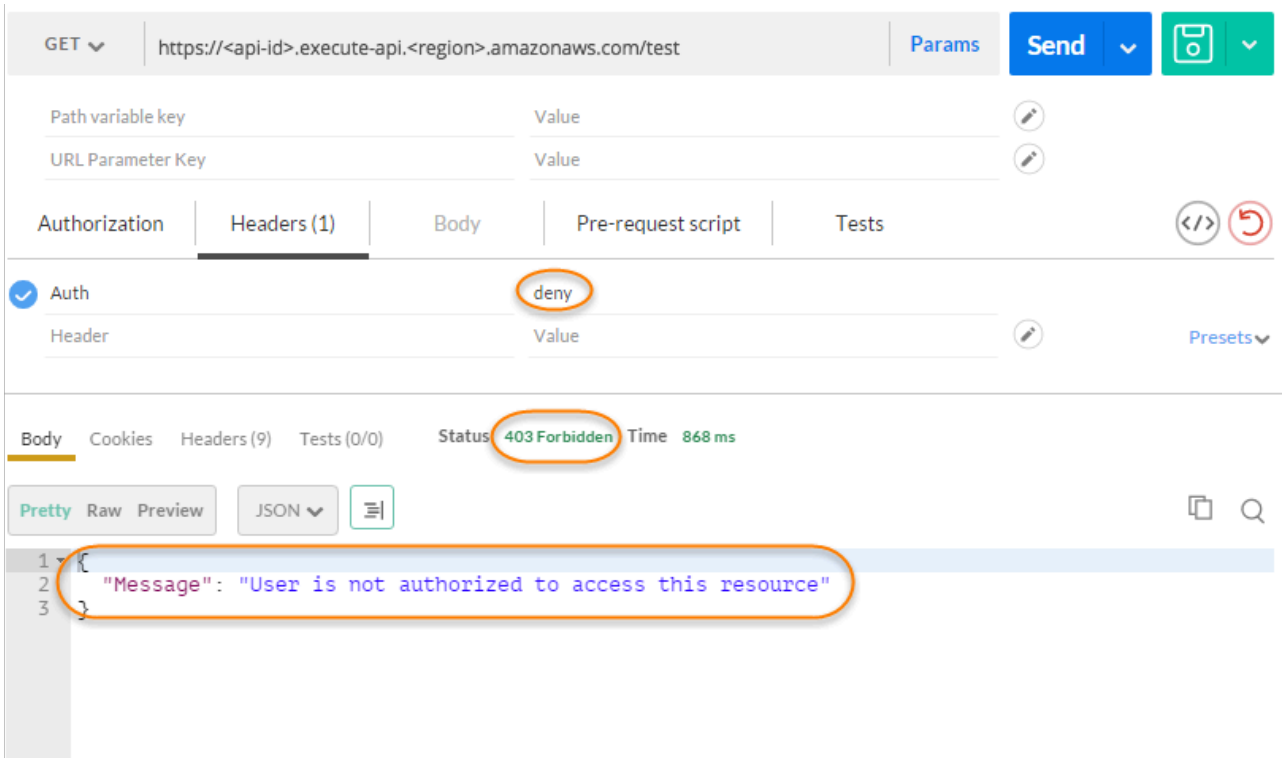
1. Apri Postman, scegli il GETmetodo e incolla Invoke URL nel campo adiacente. API URL

   Aggiungi l'intestazione del token di autorizzazione Lambda e imposta il valore su `allow`. Scegliere Send (Invia).



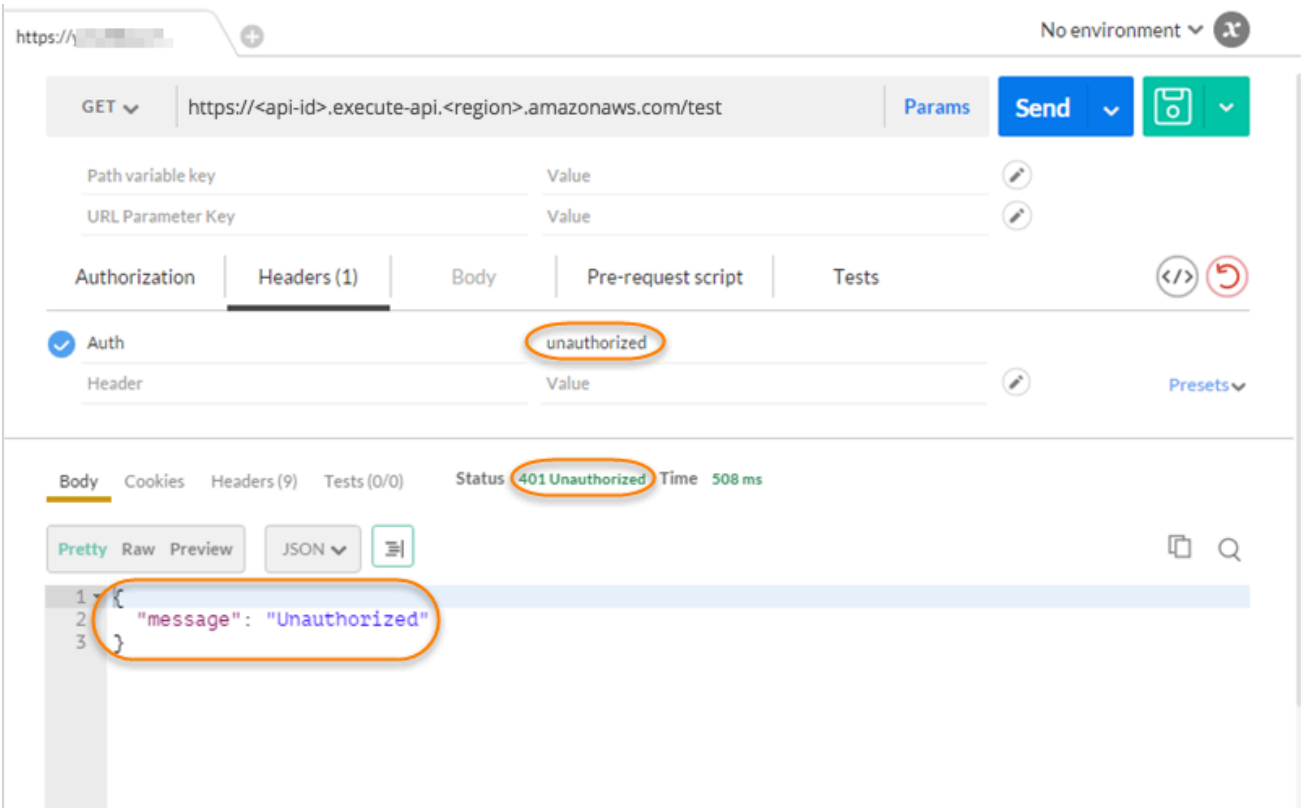
La risposta mostra che l'autorizzatore API Gateway Lambda restituisce una risposta di 200 OK e autorizza correttamente la chiamata per accedere all'HTTPendpoint (http://httpbin.org/get) integrato con il metodo.

2. Sempre in Postman, modifica il valore dell'intestazione del token di autorizzazione Lambda impostandolo su `deny`. Scegliere Send (Invia).



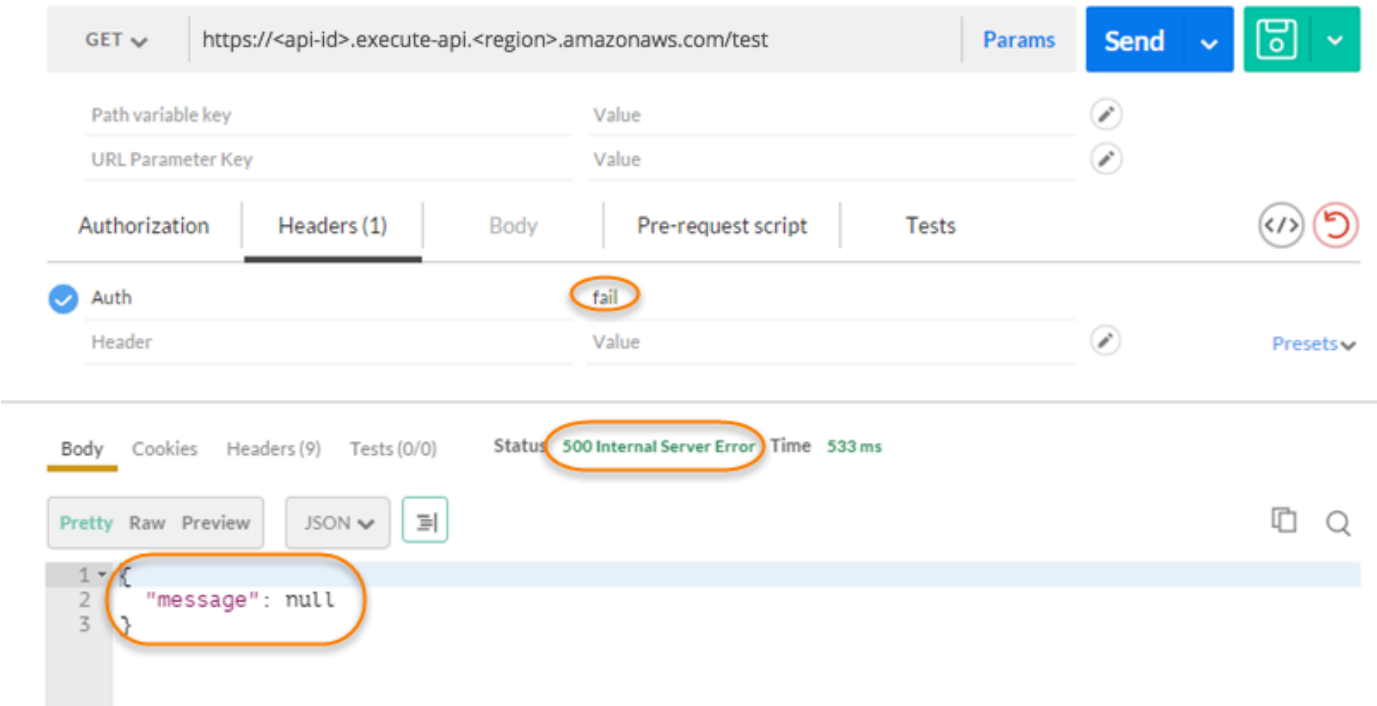
La risposta mostra che l'autorizzatore API Gateway Lambda restituisce una risposta 403 Forbidden senza autorizzare la chiamata per accedere all'endpoint. HTTP

3. In Postman, modifica il valore dell'intestazione del token di autorizzazione Lambda impostandolo su `unauthorized` e seleziona Send (Invia).



La risposta mostra che API Gateway restituisce una risposta 401 non autorizzata senza autorizzare la chiamata ad accedere all'endpoint. HTTP

4. Modifica quindi il valore dell'intestazione del token di autorizzazione Lambda impostandolo su `fail`. Scegliere Send (Invia).



La risposta mostra che API Gateway restituisce una risposta 500 Internal Server Error senza autorizzare la chiamata ad accedere all'HTTPendpoint.

Configurare un autorizzatore API Gateway Lambda per più account

Ora puoi anche utilizzare una AWS Lambda funzione di un AWS account diverso come funzione di autorizzazione. API Ogni account può trovarsi in qualsiasi regione in cui Amazon API Gateway è disponibile. La funzione di autorizzazione Lambda può utilizzare strategie di autenticazione con token portatori come o. OAuth SAML Ciò semplifica la gestione e la condivisione centralizzata di una funzione di autorizzazione Lambda centrale su più gateway. API APIs

In questa sezione, mostriamo come configurare una funzione di autorizzazione Lambda per più account utilizzando la console Amazon API Gateway.

Queste istruzioni presuppongono che tu abbia già un API Gateway API in un AWS account e una funzione di autorizzazione Lambda in un altro account.

Configurare un autorizzatore Lambda per più account utilizzando la console Gateway API

Accedi alla console Amazon API Gateway con l'account che contiene il tuoAPI, quindi procedi come segue:

1. Scegli il tuoAPI, quindi nel pannello di navigazione principale scegli Autorizzatori.

2. Scegli Create Authorizer (Crea autorizzazioni).
3. In Nome del provider di autorizzazioni, immetti il nome di un provider di autorizzazioni.
4. In Tipo di autorizzazione, seleziona Lambda.
5. Per Lambda Function, inserisci il valore completo ARN per la funzione di autorizzazione Lambda che hai nel tuo secondo account.

> **Note**
>
> Nella console Lambda, puoi trovare la funzione ARN per la tua funzione nell'angolo in alto a destra della finestra della console.

6. Verrà visualizzato un avviso con una stringa del comando `aws lambda add-permission`. Questa politica concede a API Gateway l'autorizzazione a richiamare la funzione di autorizzazione Lambda. Copiare il comando e salvarlo per un secondo momento. È possibile eseguire il comando dopo aver creato l'autorizzatore.
7. Lascia vuoto il ruolo di richiamo Lambda per consentire alla console API Gateway di impostare una policy basata sulle risorse. La politica concede a API Gateway l'autorizzazione a richiamare la funzione di autorizzazione Lambda. Puoi anche scegliere di inserire un IAM ruolo per consentire a API Gateway di richiamare la funzione di autorizzazione Lambda. Per un esempio di ruolo, consulta Creazione di un ruolo IAM assumibile.
8. In Payload evento Lambda, scegli Token per una funzione di autorizzazione basata su `TOKEN` o Richiesta per una funzione di autorizzazione basata su `REQUEST`.
9. A seconda dell'opzione scelta in precedenza, esegui una di queste operazioni:

   a. Per l'opzione Token esegui le operazioni indicate di seguito:

      - In Origine token, immetti il nome dell'intestazione contenente il token di autorizzazione. Il API client deve includere un'intestazione con questo nome per inviare il token di autorizzazione all'autorizzatore Lambda.
      - Facoltativamente, per la convalida del token, inserisci una dichiarazione. RegEx APIGateway esegue la convalida iniziale del token di input rispetto a questa espressione e richiama l'autorizzatore dopo la convalida riuscita. Questo aiuta a ridurre le chiamate al tuo. API
      - Per memorizzare nella cache la policy di autorizzazione generata dalla funzione di autorizzazione, attiva l'opzione Autorizzazione caching. Quando la memorizzazione nella cache delle policy è abilitata, puoi scegliere di modificare il TTLvalore. L'impostazione

TTLsu zero disabilita la memorizzazione nella cache delle policy. Quando la memorizzazione della policy nella cache è abilitata, il nome dell'intestazione specificato in Origine token diventa la chiave della cache. Se nella richiesta vengono passati più valori a questa intestazione, tutti i valori diventeranno la chiave della cache, con l'ordine mantenuto.

> **Note**
>
> Il TTLvalore predefinito è 300 secondi. Il valore massimo è 3600 secondi e non è possibile aumentare questo limite.

b. Per l'opzione Request (Richiesta), esegui queste operazioni:

- In Tipo di origine identità, seleziona un tipo di parametro. I tipi di parametri supportati sono `Header`, `Query string`, `Stage variable` e `Context`. Per aggiungere altre origini di identità, scegli Aggiungi parametro.
- Per memorizzare nella cache la policy di autorizzazione generata dalla funzione di autorizzazione, attiva l'opzione Autorizzazione caching. Quando la memorizzazione nella cache delle policy è abilitata, è possibile scegliere di modificare il TTLvalore. L'impostazione TTLsu zero disabilita la memorizzazione nella cache delle policy.

  APIGateway utilizza le fonti di identità specificate come chiave di memorizzazione nella cache dell'autorizzazione della richiesta. Quando la memorizzazione nella cache è abilitata, API Gateway chiama la funzione Lambda dell'autorizzatore solo dopo aver verificato con successo che tutte le fonti di identità specificate siano presenti in fase di esecuzione. Se una fonte di identificazione specificata è mancante, nulla o vuota, API Gateway restituisce una `401 Unauthorized` risposta senza chiamare la funzione Lambda dell'autorizzazione.

  Quando sono definite più origini di identità, vengono utilizzate tutte per derivare la chiave cache delle autorizzazioni. Se vengono modificate parti della chiave cache, le autorizzazioni rimuovono il documento di policy memorizzato nella cache e ne generano uno nuovo. Se nella richiesta viene passata un'intestazione con più valori, tutti i valori diventeranno la chiave della cache, con l'ordine mantenuto.
- Quando la memorizzazione nella cache è disattivata, non è necessario specificare un'origine di identità.

> **Note**
>
> Per abilitare la memorizzazione nella cache, l'autorizzatore deve restituire una politica applicabile a tutti i metodi di un. API Per applicare policy specifiche del metodo, puoi disattivare l'opzione Autorizzazione caching.

10. Scegli Create Authorizer (Crea autorizzazioni).
11. Incolla la stringa di `aws lambda add-permission` comando che hai copiato in un passaggio precedente in una AWS CLI finestra configurata per il tuo secondo account. Sostituire `AUTHORIZER_ID` con l'ID del proprio autorizzatore. In questo modo, al primo account verrà concesso l'accesso alla funzione di autorizzazione Lambda del secondo account.

Controlla l'accesso in base agli attributi di un'identità con autorizzazioni verificate

Usa Amazon Verified Permissions per controllare l'accesso al tuo API gatewayAPI. Quando utilizzi API Gateway con autorizzazioni verificate, Verified Permissions crea un sistema di autorizzazione Lambda che utilizza decisioni di autorizzazione granulari per controllare l'accesso al tuo. API Verified Permissions autorizza i chiamanti in base a uno schema e a policy store utilizzando il linguaggio di policy Cedar per definire autorizzazioni dettagliate per gli utenti dell'applicazione. Per ulteriori informazioni, consulta Creare un archivio di policy con un provider connesso API e di identità nella Amazon Verified Permissions User Guide.

Verified Permissions supporta i pool di utenti di Amazon Cognito o i provider di identità OpenID OIDC Connect () come fonti di identità. Verified Permissions presuppone che il principale sia stato precedentemente identificato e autenticato. Le autorizzazioni verificate sono supportate solo per le versioni regionali e ottimizzate per i dispositivi periferici. REST APIs

Crea un autorizzatore Lambda utilizzando le autorizzazioni verificate

Verified Permissions crea un autorizzatore Lambda per determinare se un principale è autorizzato a eseguire un'azione sul tuo. API Crei la politica Cedar che Verified Permissions utilizza per eseguire le sue attività di autorizzazione.

Di seguito è riportato un esempio di policy Cedar che consente l'accesso per richiamare un file API basato sul pool di utenti di Amazon Cognito`us-east-1_ABC1234`, per `developer` il gruppo sulla `GET /users` risorsa di un. API Le autorizzazioni verificate determinano l'appartenenza al gruppo analizzando l'identità del chiamante nel token del portatore.

```
permit(
  principal in MyAPI::UserGroup::"us-east-1_ABC1234|developer",
  action in [ MyAPI::Action::"get /users" ],
  resource
  );
```

Facoltativamente, Verified Permissions può associare l'autorizzatore ai metodi del tuo. API Nelle fasi di produzione del tuo prodottoAPI, ti consigliamo di non consentire a Verified Permissions di allegare l'autorizzatore al posto tuo.

L'elenco seguente mostra come configurare le autorizzazioni verificate per allegare o meno l'autorizzatore Lambda alla richiesta del metodo dei API propri metodi.

Allega l'autorizzatore per te ()AWS Management Console

Quando scegli Crea archivio di politiche nella console Autorizzazioni verificate, nella pagina di integrazione dell'app Deploy, scegli Ora.

Non allegare l'autorizzatore al posto tuo ()AWS Management Console

Quando scegli Crea policy store nella console Autorizzazioni verificate, nella pagina di integrazione dell'app Deploy, scegli Più tardi.

Verified Permissions crea comunque un autorizzatore Lambda per te. L'autorizzatore Lambda inizia con. `AVPAuthorizerLambda-` Per ulteriori istruzioni su come collegare l'autorizzatore a un metodo, consulta. the section called “Configurare un metodo per utilizzare un autorizzatore Lambda (console)”

Allega l'autorizzatore per te ()AWS CloudFormation

Nel AWS CloudFormation modello generato da Autorizzazioni verificate, nella `Conditions` sezione, imposta su. `"Ref": "shouldAttachAuthorizer"` `true`

Non allegare l'autorizzatore al posto tuo ()AWS CloudFormation

Nel AWS CloudFormation modello generato da Autorizzazioni verificate, nella `Conditions` sezione, imposta su. `"Ref": "shouldAttachAuthorizer"` `false`

Verified Permissions crea comunque un autorizzatore Lambda per te. L'autorizzatore Lambda inizia con. `AVPAuthorizerLambda-` Per ulteriori istruzioni su come collegare l'autorizzatore a un metodo, consulta. the section called “Configurare un metodo per utilizzare un autorizzatore Lambda ()AWS CLI”

Chiama un autorizzatore Lambda utilizzando Autorizzazioni verificate

Puoi chiamare il tuo sistema di autorizzazione Lambda fornendo un'identità o un token di accesso nell'intestazione. `Authorization` Per ulteriori informazioni, consulta the section called “Chiama un utente API con autorizzatori Lambda”.

APIGateway memorizza nella cache la policy restituita dall'autorizzatore Lambda per 120 secondi. È possibile modificarla TTL nella console API Gateway o utilizzando. AWS CLI

Controlla l'accesso alle API REST utilizzando i pool di utenti di Amazon Cognito come autorizzatore

In alternativa all'utilizzo di ruoli e policy IAM o autorizzazioni Lambda (note in precedenza come autorizzazioni ad hoc), puoi utilizzare un pool di utenti di Amazon Cognito per controllare chi può accedere all'API in Amazon API Gateway.

Per usare un pool di utenti di Amazon Cognito con l'API, devi prima creare l'autorizzazione di tipo `COGNITO_USER_POOLS` e quindi configurare un metodo API per usare tale autorizzazione. Una volta distribuita l'API, il client deve innanzitutto registrare l'utente nel pool di utenti, ottenere un token di identità o di accesso per l'utente e quindi chiamare il metodo API con uno dei token, che sono in genere impostati sull'intestazione `Authorization` della richiesta. La chiamata API riesce solo se viene fornito il token necessario e se questo è valido. In caso contrario, il client non è autorizzato ad effettuare la chiamata perché non ha credenziali che è stato possibile autorizzare.

Il token di identità viene usato per autorizzare chiamate API in base alle richieste di identità dell'utente connesso. Il token di accesso viene usato per autorizzare chiamate API in base agli ambiti personalizzati di risorse con accesso protetto specificate. Per ulteriori informazioni, consulta Utilizzo di token con pool di utenti e la pagina relativa a server di risorse e ambiti personalizzati.

Per creare e configurare un pool di utenti di Amazon Cognito per l'API, esegui le attività seguenti:

- Usa la console Amazon Cognito, CLI/SDK o API per creare un pool di utenti o usane uno di proprietà di un altro account. AWS
- Usa la console, l'interfaccia a riga di comando/SDK o l'API di API Gateway, per creare autorizzazioni API Gateway con il pool di utenti scelto.
- Usa la console, l'interfaccia a riga di comando/SDK o l'API di API Gateway, per abilitare le autorizzazioni per i metodi API selezionati.

Per chiamare qualsiasi metodo API con un pool di utenti abilitato, i client API eseguono le attività seguenti:

- Utilizzano l'interfaccia a riga di comando/SDK o l'API di Amazon Cognito per fare accedere un utente al pool di utenti scelto e ottenere un token di identità o di accesso. Per ulteriori informazioni sull'utilizzo degli SDK, consulta Esempi di codice per Amazon Cognito AWS utilizzando gli SDK.
- Utilizzano un framework specifico del client per chiamare l'API di API Gateway distribuita e specificare il token appropriato nell'intestazione `Authorization`.

In quanto sviluppatore dell'API, devi fornire agli sviluppatori client l'ID pool di utenti, un ID client e possibilmente i segreti client definiti come parte del pool di utenti.

> **Note**
>
> Per permettere a un utente di accedere usando credenziali Amazon Cognito e anche di ottenere credenziali temporanee da usare con le autorizzazioni di un ruolo IAM, usa identità federate di Amazon Cognito. Per ogni metodo HTTP dell'endpoint della risorsa API, imposta il tipo di autorizzazione, la categoria `Method Execution`, su `AWS_IAM`.

In questa sezione descriveremo come creare un pool di utenti, come integrare un'API di API Gateway con il pool di utenti e come richiamare un'API integrata con il pool di utenti.

Argomenti

- Ottenimento delle autorizzazioni per creare le autorizzazioni per il pool di utenti di Amazon Cognito per un'API REST
- Creazione di un pool di utenti di Amazon Cognito per un'API REST
- Integrazione di un'API REST con un pool di utenti di Amazon Cognito
- Chiamata di un'API REST integrata con un pool di utenti di Amazon Cognito
- Configurazione dell'autorizzazione Amazon Cognito tra account per un'API REST tramite la console API Gateway
- Crea un autorizzatore Amazon Cognito per un'API REST utilizzando AWS CloudFormation

Ottenimento delle autorizzazioni per creare le autorizzazioni per il pool di utenti di Amazon Cognito per un'API REST

Per creare un'autorizzazione con un pool di utenti di Amazon Cognito, devi disporre di autorizzazioni `Allow` per creare o aggiornare un'autorizzazione con il pool di utenti di Amazon Cognito scelto. Il documento di policy IAM seguente mostra un esempio di queste autorizzazioni:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "apigateway:POST"
            ],
            "Resource": "arn:aws:apigateway:*::/restapis/*/authorizers",
            "Condition": {
                "ArnLike": {
                    "apigateway:CognitoUserPoolProviderArn": [
                        "arn:aws:cognito-idp:us-east-1:123456789012:userpool/us-
east-1_aD06NQmjO",
                        "arn:aws:cognito-idp:us-east-1:234567890123:userpool/us-
east-1_xJ1MQtPEN"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "apigateway:PATCH"
            ],
            "Resource": "arn:aws:apigateway:*::/restapis/*/authorizers/*",
            "Condition": {
                "ArnLike": {
                    "apigateway:CognitoUserPoolProviderArn": [
                        "arn:aws:cognito-idp:us-east-1:123456789012:userpool/us-
east-1_aD06NQmjO",
                        "arn:aws:cognito-idp:us-east-1:234567890123:userpool/us-
east-1_xJ1MQtPEN"
                    ]
                }
            }
```

```
        }
    ]
}
```

Assicurarsi che la policy sia collegata a un gruppo IAM cui l'utente appartiene o a un ruolo IAM assegnato.

Nel documento di policy precedente l'operazione `apigateway:POST` è per la creazione di nuove autorizzazioni, mentre l'operazione `apigateway:PATCH` è per l'aggiornamento di autorizzazioni esistenti. Puoi limitare la policy a una regione specifica o a una determinata API sostituendo, rispettivamente, i primi due caratteri jolly (*) dei valori di `Resource`.

Le clausole `Condition` usate qui sono per limitare le autorizzazioni `Allowed` ai pool di utenti specificati. Quando è presente una clausola `Condition`, l'accesso a uno dei pool di utenti che non corrispondono alle condizioni viene negato. Quando un'autorizzazione non include una clausola `Condition` è permesso l'accesso a qualsiasi pool di utenti.

Per impostare la clausola `Condition`, sono disponibili le opzioni seguenti:

- Puoi impostare un'espressione condizionale `ArnLike` o `ArnEquals` per permettere la creazione o l'aggiornamento di autorizzazioni `COGNITO_USER_POOLS` solo con i pool di utenti specificati.
- Puoi impostare un'espressione condizionale `ArnNotLike` o `ArnNotEquals` per permettere la creazione o l'aggiornamento di autorizzazioni `COGNITO_USER_POOLS` con qualsiasi pool di utenti non specificato nell'espressione.
- Puoi omettere la clausola `Condition` per permettere la creazione o l'aggiornamento di autorizzazioni `COGNITO_USER_POOLS` con qualsiasi pool di utenti di qualsiasi account AWS e in qualsiasi regione.

Per ulteriori informazioni sulle espressioni condizionali per gli Amazon Resource Name (ARN), consulta la pagina Amazon relativa agli operatori di condizione per i nomi di risorsa. Come mostrato nell'esempio, `apigateway:CognitoUserPoolProviderArn` è un elenco di ARN dei pool di utenti `COGNITO_USER_POOLS` che possono o non possono essere usati con autorizzazioni API Gateway di tipo `COGNITO_USER_POOLS`.

Creazione di un pool di utenti di Amazon Cognito per un'API REST

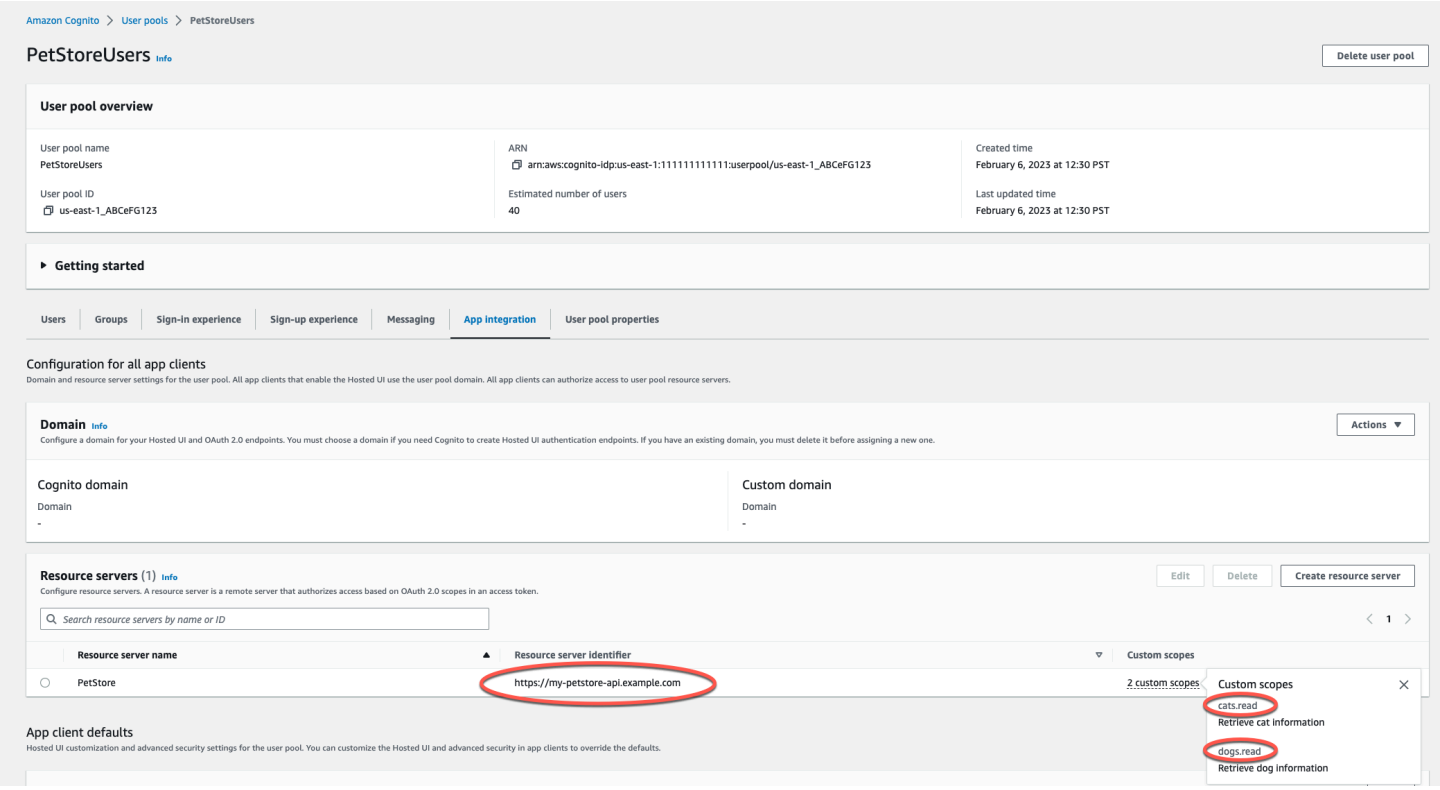
Prima di integrare l'API con un pool di utenti, devi creare il pool di utenti in Amazon Cognito. La configurazione del pool di utenti deve rispettare tutte le quote di risorse per Amazon Cognito. Tutte

le variabili Amazon Cognito definite dall'utente, come gruppi, utenti e ruoli, devono utilizzare solo caratteri alfanumerici. Per istruzioni su come creare un pool di utenti, consulta Tutorial: Creazione di un bacino d'utenza nella Guida per gli sviluppatori di Amazon Cognito.

Prendi nota dell'ID pool di utenti, dell'ID client e di qualsiasi segreto client. Il client dovrà fornirli ad Amazon Cognito perché l'utente possa registrarsi con il pool di utenti, accedere al pool di utenti e ottenere un token di identità o di accesso da includere nelle richieste per chiamare metodi API configurati con il pool di utenti. Inoltre, devi specificare il nome del pool di utenti quando configuri il pool di utenti come autorizzazioni in API Gateway, come descritto nella prossima sezione.

Se stai usando token di accesso per autorizzare le chiamate di metodi API, assicurati di configurare l'integrazione dell'app con il pool di utenti per configurare gli ambiti personalizzati che desideri in un determinato server di risorse. Per ulteriori informazioni sull'utilizzo di token con pool di utenti di Amazon Cognito, consulta Utilizzo di token con pool di utenti. Per ulteriori informazioni sui server di risorse, consulta Definizione dei server di risorse per il pool di utenti.

Annota gli identificatori dei server di risorse configurati e i nomi degli ambiti personalizzati. Ti serviranno per creare i nomi completi degli ambiti di accesso per l'opzione OAuth Scopes (Ambiti OAuth), che viene utilizzata dal provider di autorizzazioni `COGNITO_USER_POOLS`.



Integrazione di un'API REST con un pool di utenti di Amazon Cognito

Dopo aver creato un pool di utenti di Amazon Cognito in API Gateway, devi creare un'autorizzazione `COGNITO_USER_POOLS` che usa il pool di utenti. La procedura seguente illustra come eseguire questa operazione tramite la console API Gateway.

> **Note**
>
> Puoi utilizzare l'operazione `CreateAuthorizer` per creare un'autorizzazione `COGNITO_USER_POOLS` che utilizzi più pool di utenti. Puoi utilizzare fino a 1.000 pool di utenti per un unica autorizzazione `COGNITO_USER_POOLS`. Questo limite non può essere aumentato.

> **Important**
>
> Dopo aver eseguito una delle procedure di seguito, è necessario distribuire o ridistribuire l'API per la propagazione delle modifiche. Per ulteriori informazioni sulla distribuzione della tua API, vedi Implementazione REST APIs in Gateway API.

Per creare un'autorizzazione **`COGNITO_USER_POOLS`** tramite la console API Gateway

1. Crea una nuova API oppure selezionane una esistente in API Gateway.
2. Nel riquadro di navigazione principale, scegli Autorizzazioni.
3. Scegli Create Authorizer (Crea autorizzazioni).
4. Per configurare la nuova autorizzazione per usare un pool di utenti, esegui queste operazioni:
   a. In Nome del provider di autorizzazioni, immetti un nome.
   b. In Tipo di autorizzazione, seleziona Cognito.
   c. Per il pool di utenti Cognito, scegli Regione AWS dove hai creato Amazon Cognito e seleziona un pool di utenti disponibile.

      Puoi utilizzare una variabile di fase per definire il tuo pool di utenti. Usa il seguente formato per il tuo pool di utenti: `arn:aws:cognito-idp:us-east-2:111122223333:userpool/${stageVariables.MyUserPool}`.

d. In Origine token, immetti **`Authorization`** come nome di intestazione da passare al token di identità o di accesso restituito da Amazon Cognito quando un utente accede correttamente.

e. (Facoltativo) Immetti un'espressione regolare nel campo Convalida del token per convalidare il campo `aud` (Audience, Destinatari) del token di identità prima che la richiesta venga autorizzata con Amazon Cognito. Si noti che quando si utilizza un token di accesso questa convalida rifiuta la richiesta poiché il token di accesso non contiene il campo `aud`.

f. Scegli Create Authorizer (Crea autorizzazioni).

5. Dopo aver creato l'autorizzazione `COGNITO_USER_POOLS`, puoi facoltativamente testare la chiamata specificando un token di identità assegnato dal pool di utenti. Puoi ottenere questo token di identità chiamando l'SDK Amazon Cognito Identity per eseguire l'accesso dell'utente. È anche possibile usare l'operazione `InitiateAuth`. Se in Ambiti di autorizzazione non configuri alcun valore, Gateway API considera il token fornito come un token di identità.

La procedura precedente crea un'autorizzazione `COGNITO_USER_POOLS` che utilizza il nuovo pool di utenti di Amazon Cognito appena creato. A seconda del modo in cui abiliti l'autorizzazione per un metodo API, puoi usare un token di identità o un token di accesso assegnato dal pool di utenti integrato.

Per configurare un'autorizzazione **`COGNITO_USER_POOLS`** per i metodi

1. Scegliere Resources (Risorse). Seleziona un nuovo metodo o scegline uno esistente. Se necessario, crea una risorsa.
2. Nella scheda Richiesta metodo, in Impostazioni richiesta metodo, scegli Modifica.
3. In Autorizzazioni, nel menu a discesa, seleziona Autorizzazioni del gruppo di utenti di Cognito.
4. Per usare un token di identità, esegui queste operazioni:

   a. Non specificare alcun valore in Ambiti di autorizzazione.

   b. Se necessario, in Richiesta di integrazione, aggiungi le espressioni `$context.authorizer.claims['property-name']` o `$context.authorizer.claims.property-name` in un modello di mappatura del corpo per passare la proprietà delle richieste di identità specificata dal pool di utenti al back-end. Per i nomi di proprietà semplici, come `sub` o `custom-sub`, le due notazioni sono identiche. Per i nomi di proprietà complessi, come `custom:role`, non puoi usare la notazione punto.

Ad esempio, le espressioni di mappatura seguenti passano i campi standard `sub` e `email` della richiesta al back-end:

```
{
 "context" : {
  "sub" : "$context.authorizer.claims.sub",
  "email" : "$context.authorizer.claims.email"
 }
}
```

Se hai dichiarato un campo di richiesta personalizzato quando hai configurato un pool di utenti, puoi seguire lo stesso modello per accedere ai campi personalizzati. L'esempio seguente ottiene un campo `role` personalizzato di una richiesta:

```
{
 "context" : {
  "role" : "$context.authorizer.claims.role"
    }
}
```

Se il campo personalizzato della richiesta viene dichiarato come `custom:role`, usa l'esempio seguente per ottenere le proprietà della richiesta:

```
{
 "context" : {
  "role" : "$context.authorizer.claims['custom:role']"
    }
}
```

5. Per usare un token di accesso, esegui queste operazioni:

   a. In Ambiti di autorizzazione, immetti uno o più nomi completi di un ambito configurato quando è stato creato il pool di utenti di Amazon Cognito. Ad esempio, seguendo l'esempio fornito in Creazione di un pool di utenti di Amazon Cognito per un'API REST, uno degli ambiti è `https://my-petstore-api.example.com/cats.read`.

      In fase di runtime, la chiamata del metodo riesce se qualsiasi ambito specificato nel metodo in questa fase corrisponde a un ambito richiesto nel token in ingresso. Altrimenti, la chiamata non riesce e restituisce una risposta `401 Unauthorized`.

   b. Selezionare Salva.

6. Ripeti queste fasi per gli altri metodi scelti.

Se con l'autorizzazione `COGNITO_USER_POOLS` l'opzione OAuth Scopes (Ambiti OAuth) non è specificata, API Gateway considera il token specificato un token di identità e verifica l'identità richiesta rispetto a quella del pool di utenti. Altrimenti, API Gateway considera il token specificato un token di accesso e verifica gli ambiti di accesso richiesti nel token rispetto agli ambiti di autorizzazione dichiarati nel metodo.

Invece di usare la console API Gateway, puoi anche abilitare un pool di utenti di Amazon Cognito in un metodo specificando un file di definizione OpenAPI e importando la definizione API in API Gateway.

Per importare un'autorizzazione COGNITO_USER_POOLS con un file di definizione OpenAPI

1. Crea (o esporta) un file di definizione OpenAPI per l'API.
2. Specificare la definizione JSON dell'autorizzazione `COGNITO_USER_POOLS` (`MyUserPool`) come parte della sezione `securitySchemes` OpenAPI 3.0 o della sezione `securityDefinitions` in OpenAPI 2.0, nel seguente modo as follows:

   OpenAPI 3.0

   ```
   "securitySchemes": {
     "MyUserPool": {
       "type": "apiKey",
       "name": "Authorization",
       "in": "header",
       "x-amazon-apigateway-authtype": "cognito_user_pools",
       "x-amazon-apigateway-authorizer": {
         "type": "cognito_user_pools",
         "providerARNs": [
           "arn:aws:cognito-idp:{region}:{account_id}:userpool/{user_pool_id}"
         ]
       }
     }
   ```

   OpenAPI 2.0

   ```
   "securityDefinitions": {
     "MyUserPool": {
       "type": "apiKey",
   ```

```
      "name": "Authorization",
      "in": "header",
      "x-amazon-apigateway-authtype": "cognito_user_pools",
      "x-amazon-apigateway-authorizer": {
        "type": "cognito_user_pools",
        "providerARNs": [
          "arn:aws:cognito-idp:{region}:{account_id}:userpool/{user_pool_id}"
        ]
      }
    }
```

3. Per usare il token di identità per l'autorizzazione del metodo, aggiungi `{ "MyUserPool": [] }` alla definizione `security` del metodo, come mostrato nel metodo GET seguente nella risorsa root.

```
"paths": {
  "/": {
    "get": {
      "consumes": [
        "application/json"
      ],
      "produces": [
        "text/html"
      ],
      "responses": {
        "200": {
          "description": "200 response",
          "headers": {
            "Content-Type": {
              "type": "string"
            }
          }
        }
      },
      "security": [
        {
          "MyUserPool": []
        }
      ],
      "x-amazon-apigateway-integration": {
        "type": "mock",
        "responses": {
          "default": {
```

```
              "statusCode": "200",
              "responseParameters": {
                "method.response.header.Content-Type": "'text/html'"
              },
            }
          },
          "requestTemplates": {
            "application/json": "{\"statusCode\": 200}"
          },
          "passthroughBehavior": "when_no_match"
        }
      },
      ...
  }
```

4. Per usare il token di accesso per l'autorizzazione del metodo, modifica la definizione di sicurezza precedente in `{ "MyUserPool": [resource-server/scope, ...] }`:

```
  "paths": {
    "/": {
      "get": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "text/html"
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "headers": {
              "Content-Type": {
                "type": "string"
              }
            }
          }
        },
        "security": [
          {
            "MyUserPool": ["https://my-petstore-api.example.com/cats.read",
"http://my.resource.com/file.read"]
          }
        ],
```

```
        "x-amazon-apigateway-integration": {
          "type": "mock",
          "responses": {
            "default": {
              "statusCode": "200",
              "responseParameters": {
                "method.response.header.Content-Type": "'text/html'"
              },
            }
          },
          "requestTemplates": {
            "application/json": "{\"statusCode\": 200}"
          },
          "passthroughBehavior": "when_no_match"
        }
      },
      ...
    }
```

5. Se necessario, è possibile definire altre impostazioni di configurazione dell'API utilizzando le definizioni o estensioni OpenAPI appropriate. Per ulteriori informazioni, consulta APIEstensioni aperte per API Gateway.

Chiamata di un'API REST integrata con un pool di utenti di Amazon Cognito

Per chiamare un metodo con un'autorizzazione del pool di utenti configurata, il client deve eseguire queste operazioni:

- Permettere all'utente di iscriversi al pool di utenti.
- Permettere all'utente di accedere al pool di utenti.
- Ottenere un token di identità o di accesso per l'utente che ha effettuato l'accesso dal pool di utenti.
- Includere il token nell'intestazione `Authorization` (o in un'altra intestazione specificata durante la creazione dell'autorizzazione).

È possibile utilizzare AWS Amplify per eseguire queste attività. Per maggiori informazioni, consulta Integrazione di Amazon Cognito con le app Web e per dispositivi mobili.

- Per Android, consulta Nozioni di base su Amplify per Android.
- Per usare iOS, consulta Nozioni di base su Amplify per iOS.

- Per utilizzarlo JavaScript, consulta Getting Started with Amplify for Javascript.

Configurazione dell'autorizzazione Amazon Cognito tra account per un'API REST tramite la console API Gateway

Ora puoi anche utilizzare un pool di utenti Amazon Cognito di un altro AWS account come autorizzatore API. Il pool di utenti di Amazon Cognito può utilizzare strategie di autenticazione con token di connessione come OAuth o SAML. In questo modo, è più semplice gestire e condividere a livello centralizzato l'autorizzazione di un pool di utenti di Amazon Cognito centrale tra più API di API Gateway.

In questa sezione, viene illustrato come configurare un pool di utenti di Amazon Cognito tra account tramite la console Amazon API Gateway.

Queste istruzioni presuppongono che tu abbia già un'API API Gateway in un AWS account e un pool di utenti Amazon Cognito in un altro account.

Crea un autorizzatore Amazon Cognito per più account per un'API REST

Accedi alla console Gateway Amazon API con l'account contenente la tua API e quindi procedi come descritto di seguito:

1. Crea una nuova API oppure selezionane una esistente in API Gateway.
2. Nel riquadro di navigazione principale, scegli Autorizzazioni.
3. Scegli Create Authorizer (Crea autorizzazioni).
4. Per configurare la nuova autorizzazione per usare un pool di utenti, esegui queste operazioni:
   a. In Nome del provider di autorizzazioni, immetti un nome.
   b. In Tipo di autorizzazione, seleziona Cognito.
   c. Per Gruppo di utenti di Cognito, copia e incolla l'ARN completo del pool di utenti nel secondo account.

> **Note**
> Nella console Amazon Cognito, puoi trovare l'ARN del pool di utenti nel campo Pool ARN (ARN pool) del riquadro General Settings (Impostazioni generali).

d. In Origine token, immetti **`Authorization`** come nome di intestazione da passare al token di identità o di accesso restituito da Amazon Cognito quando un utente accede correttamente.

e. (Facoltativo) Immetti un'espressione regolare nel campo Convalida del token per convalidare il campo `aud` (Audience, Destinatari) del token di identità prima che la richiesta venga autorizzata con Amazon Cognito. Si noti che quando si utilizza un token di accesso questa convalida rifiuta la richiesta poiché il token di accesso non contiene il campo `aud`.

f. Scegli Create Authorizer (Crea autorizzazioni).

Crea un autorizzatore Amazon Cognito per un'API REST utilizzando AWS CloudFormation

Puoi utilizzarlo AWS CloudFormation per creare un pool di utenti Amazon Cognito e un autorizzatore Amazon Cognito. Il AWS CloudFormation modello di esempio esegue le seguenti operazioni:

- Crea un pool di utenti di Amazon Cognito. Il client deve prima far accedere l'utente al pool di utenti e ottenere un'identità o un token di accesso. Se stai usando token di accesso per autorizzare le chiamate di metodi API, assicurati di configurare l'integrazione dell'app con il pool di utenti per configurare gli ambiti personalizzati che desideri in un determinato server di risorse.
- Crea un'API Gateway API con un metodo `GET`.
- Crea un'autorizzazione Amazon Cognito che utilizza l'intestazione `Authorization` come origine del token.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  UserPool:
    Type: AWS::Cognito::UserPool
    Properties:
      AccountRecoverySetting:
        RecoveryMechanisms:
          - Name: verified_phone_number
            Priority: 1
          - Name: verified_email
            Priority: 2
      AdminCreateUserConfig:
        AllowAdminCreateUserOnly: true
      EmailVerificationMessage: The verification code to your new account is {####}
      EmailVerificationSubject: Verify your new account
      SmsVerificationMessage: The verification code to your new account is {####}
```

```
      VerificationMessageTemplate:
        DefaultEmailOption: CONFIRM_WITH_CODE
        EmailMessage: The verification code to your new account is {####}
        EmailSubject: Verify your new account
        SmsMessage: The verification code to your new account is {####}
    UpdateReplacePolicy: Retain
    DeletionPolicy: Retain
  CogAuthorizer:
    Type: AWS::ApiGateway::Authorizer
    Properties:
      Name: CognitoAuthorizer
      RestApiId:
        Ref: Api
      Type: COGNITO_USER_POOLS
      IdentitySource: method.request.header.Authorization
      ProviderARNs:
        - Fn::GetAtt:
            - UserPool
            - Arn
  Api:
    Type: AWS::ApiGateway::RestApi
    Properties:
      Name: MyCogAuthApi
  ApiDeployment:
    Type: AWS::ApiGateway::Deployment
    Properties:
      RestApiId:
        Ref: Api
    DependsOn:
      - CogAuthorizer
      - ApiGET
  ApiDeploymentStageprod:
    Type: AWS::ApiGateway::Stage
    Properties:
      RestApiId:
        Ref: Api
      DeploymentId:
        Ref: ApiDeployment
      StageName: prod
  ApiGET:
    Type: AWS::ApiGateway::Method
    Properties:
      HttpMethod: GET
      ResourceId:
```

```
        Fn::GetAtt:
          - Api
          - RootResourceId
      RestApiId:
        Ref: Api
      AuthorizationType: COGNITO_USER_POOLS
      AuthorizerId:
        Ref: CogAuthorizer
      Integration:
        IntegrationHttpMethod: GET
        Type: HTTP_PROXY
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets
Outputs:
  ApiEndpoint:
    Value:
      Fn::Join:
        - ""
        - - https://
          - Ref: Api
          - .execute-api.
          - Ref: AWS::Region
          - "."
          - Ref: AWS::URLSuffix
          - /
          - Ref: ApiDeploymentStageprod
          - /
```

Integrazioni per le API REST in API Gateway

Dopo aver impostato un metodo API, devi integrarlo con un endpoint nel back-end. Un endpoint di backend viene anche definito endpoint di integrazione e può essere una funzione Lambda, una pagina Web HTTP o un'azione di servizio. AWS

Come nel caso del metodo API, l'integrazione dell'API presenta una richiesta e una risposta di integrazione. Una richiesta di integrazione comprende una richiesta HTTP ricevuta dal back-end. Potrebbe o non potrebbe essere diversa dalla richiesta di metodo inviata dal cliente. Una risposta di integrazione è una risposta HTTP contenente l'output restituito dal back-end.

La configurazione di una richiesta di integrazione comporta le seguenti operazioni: configurare come passare le richieste di metodo inviate dal client al back-end; configurare come trasformare i dati della richiesta, se necessario, in dati della richiesta di integrazione; specificare quale funzione Lambda

chiamare; specificare a quale server HTTP inoltrare la richiesta in arrivo o specificare l'azione del servizio AWS da chiamare.

La configurazione di una risposta di integrazione (applicabile solo alle integrazioni non proxy) implica le seguenti operazioni: come passare il risultato restituito dal back-end alla risposta di un metodo di un dato codice di stato, definire come trasformare parametri di risposta di integrazione specifici in base a parametri di risposta dei metodi pre-configurati e configurare come mappare il corpo della risposta di integrazione al corpo della risposta del metodo in base a modelli specifici di mappatura del corpo.

A livello di programmazione, una richiesta di integrazione viene incapsulata dalla risorsa `Integration`, mentre la risposta di integrazione dalla risorsa `IntegrationResponse` di API Gateway.

Per configurare una richiesta di integrazione, è necessario creare una risorsa `Integration` e usarla per configurare l'URL dell'endpoint di integrazione. Successivamente, devi impostare le autorizzazioni IAM per accedere al back-end e specificare le mappature per trasformare i dati di richiesta in entrata prima di passarli al back-end. Per configurare una risposta di integrazione per un'integrazione non proxy, è necessario creare una risorsa `IntegrationResponse` e usarla per configurare la relativa risposta del metodo target. Successivamente, devi configurare la mappatura dell'output di back-end alla risposta del metodo.

Argomenti

- Configurazione di una richiesta di integrazione in API Gateway
- Configurazione di una risposta di integrazione in API Gateway
- Integrazioni Lambda per in Gateway REST APIs API
- Integrazioni HTTP per le API REST in API Gateway
- Integrazioni private per REST APIs in API Gateway
- Integrazioni fittizie per le API REST in API Gateway

Configurazione di una richiesta di integrazione in API Gateway

Per configurare una richiesta di integrazione, è necessario completare i seguenti task obbligatori e opzionali:

1. Seleziona un tipo di integrazione che stabilisca in che modo i dati di richiesta del metodo vengano passati al back-end.

2. Per le integrazioni non fittizie, specifica un metodo HTTP e l'URI dell'endpoint di integrazione target, eccetto per l'integrazione `MOCK`.
3. Per le integrazioni con le funzioni Lambda e AWS altre azioni di servizio, imposta un ruolo IAM con le autorizzazioni necessarie affinché API Gateway chiami il backend per tuo conto.
4. Per integrazioni non-proxy, imposta le mappature dei parametri necessari per mappare i parametri delle richieste di metodo predefinite ai parametri di richiesta di integrazione più appropriati.
5. Per integrazioni non-proxy, imposta le mappature dei corpi necessari per mappare il corpo delle richieste di metodo in entrata di un tipo di contenuto specifico in base al modello di mappatura designato.
6. Per le integrazioni non-proxy, specifica la condizione in base alla quale i dati della richiesta di metodo in entrata vengano passati attraverso il back-end inalterati.
7. A scelta, specifica come gestire una conversione dei tipi per un payload binario.
8. A scelta, dichiara il nome del namespace cache e i parametri della chiave cache per abilitare il caching delle API.

L'esecuzione di queste attività implica la creazione di una risorsa Integration di API Gateway e la configurazione di valori di proprietà adeguati. Puoi farlo utilizzando la console API Gateway, AWS CLI i comandi, un AWS SDK o l'API REST di API Gateway.

Argomenti

- Task di base di una richiesta di integrazione API
- Scegliere un tipo di integrazione API Gateway API
- Configurazione di un'integrazione proxy mediante una risorsa proxy
- Configurazione di una richiesta di integrazione API tramite la console API Gateway

Task di base di una richiesta di integrazione API

Una richiesta di integrazione è una richiesta HTTP che API Gateway invia al back-end, passando i dati della richiesta inviata dal client e trasformandoli, se necessario. Il metodo HTTP (o verbo) e l'URI della richiesta di integrazione sono determinati dal back-end (l'endpoint dell'integrazione). Possono essere diversi o uguali, rispettivamente, al metodo HTTP e all'URI della richiesta di metodo.

Ad esempio, quando una funzione Lambda restituisce un file proveniente da Amazon S3, puoi esporre intuitivamente questa operazione al client come una richiesta del metodo `GET`, anche se la corrispondente richiesta di integrazione richiede l'uso di una richiesta `POST` per chiamare la funzione

Lambda. Per un endpoint HTTP, è probabile che la richiesta di metodo e la richiesta di integrazione corrispondente utilizzino entrambi lo stesso verbo HTTP. Tuttavia, non è obbligatorio. Puoi integrare la seguente richiesta di metodo:

```
GET /{var}?query=value
Host: api.domain.net
```

Con la seguente richiesta di integrazione:

```
POST /
Host: service.domain.com
Content-Type: application/json
Content-Length: ...

{
   path: "{var}'s value",
   type: "value"
}
```

Come sviluppatore di API, puoi utilizzare qualunque verbo HTTP e URI per una richiesta di metodo che soddisfino i tuoi requisiti. Tuttavia, devi soddisfare i requisiti dell'endpoint di integrazione. Quando i dati della richiesta di metodo sono diversi dai dati della richiesta di integrazione, puoi eliminare la differenza tramite mappature dei dati della richiesta di metodo ai dati della richiesta di integrazione.

Negli esempi precedenti, la mappatura traduce i valori della variabile di percorso (`{var}`) e dei parametri di query (`query`) della richiesta di metodo `GET` nei valori delle proprietà del payload della richiesta di integrazione di `path` e `type`. Tra gli altri dati di richiesta mappabili figurano il corpo e le intestazioni della richiesta. Tali dati sono descritti in [Imposta le mappature dei dati di richiesta e risposta utilizzando la console Gateway API](#).

Quando configuri la richiesta HTTP o di integrazione proxy HTTP, assegni l'URL dell'endpoint HTTP del back-end come valore URI della richiesta di integrazione. Ad esempio, nell' PetStoreAPI, la richiesta del metodo per ottenere una pagina di animali domestici ha il seguente URI di richiesta di integrazione:

```
http://petstore-demo-endpoint.execute-api.com/petstore/pets
```

Quando configuri Lambda o l'integrazione proxy Lambda, assegni l'Amazon Resource Name (ARN) per chiamare la funzione Lambda come valore URI della richiesta di integrazione. Questo ARN ha il seguente formato:

```
arn:aws:apigateway:api-region:lambda:path//2015-03-31/functions/arn:aws:lambda:lambda-region:account-id:function:lambda-function-name/invocations
```

La parte dopo `arn:aws:apigateway:`*`api-region`*`:lambda:path/`, ovvero `/2015-03-31/functions/arn:aws:lambda:`*`lambda-region`*`:`*`account-id`*`:function:`*`lambda-function-name`*`/invocations`, è il percorso URI dell'API REST dell'azione Lambda Invoke. Se utilizzi la console API Gateway per configurare l'integrazione Lambda, API Gateway crea l'ARN e lo assegna all'URI di integrazione dopo avere chiesto di selezionare il *`lambda-function-name`* da una regione.

Quando si imposta la richiesta di integrazione con un'altra azione di AWS servizio, anche l'URI della richiesta di integrazione è un ARN, simile all'integrazione con l'azione Lambda. `Invoke` Ad esempio, per l'integrazione con l'GetBucketazione di Amazon S3, l'URI della richiesta di integrazione è un ARN del seguente formato:

```
arn:aws:apigateway:api-region:s3:path/{bucket}
```

L'URI della richiesta di integrazione è la convenzione del percorso che specifica l'operazione, dove *`{bucket}`* è il segnaposto del nome di un bucket. In alternativa, è possibile fare riferimento a un'azione di AWS servizio tramite il relativo nome. Utilizzando il nome dell'operazione, l'URI della richiesta di integrazione per l'operazione `GetBucket` di Amazon S3; diventa il seguente:

```
arn:aws:apigateway:api-region:s3:action/GetBucket
```

Con l'URI della richiesta di integrazione basata sull'operazione, il nome del bucket (*`{bucket}`*) deve essere specificato nel corpo della richiesta di integrazione (`{ Bucket: "`*`{bucket}`*`" }`), seguito dal formato di input dell'operazione `GetBucket`.

Per AWS le integrazioni, devi anche configurare le credenziali per consentire ad API Gateway di richiamare le azioni integrate. Puoi creare un ruolo IAM o sceglierne uno esistente per consentire ad API Gateway di chiamare l'operazione, quindi specificare il ruolo utilizzando il proprio ARN. Di seguito è illustrato un esempio di questo ARN:

```
arn:aws:iam::account-id:role/iam-role-name
```

Tale ruolo IAM deve contenere una policy che consenta di eseguire l'operazione. API Gateway deve inoltre essere dichiarata (nella relazione di fiducia del ruolo) come entità affidabile per l'assunzione del ruolo. Tali autorizzazioni possono essere assegnate per l'operazione stessa. Si tratta di autorizzazioni basate sulle risorse. Per l'integrazione Lambda, puoi chiamare l'operazione Lambda addPermission per impostare le autorizzazioni basate sulle risorse, quindi configurare `credentials` su null nella richiesta di integrazione di API Gateway.

Abbiamo illustrato la configurazione di integrazione di base. Le impostazioni avanzate prevedono la mappatura dei dati della richiesta di metodo ai dati della richiesta di integrazione. Dopo aver illustrato la configurazione di base di una risposta di integrazione, parliamo ora di argomenti avanzati in Imposta le mappature dei dati di richiesta e risposta utilizzando la console Gateway API, in cui affrontiamo anche lo spostamento di payload e la gestione di codifiche dei contenuti.

Scegliere un tipo di integrazione API Gateway API

Seleziona un tipo di integrazione API in base ai tipi di endpoint di integrazione con cui lavori e alla quantità di dati che transitano nell'endpoint di integrazione. Per una funzione Lambda, puoi avere l'integrazione proxy di Lambda o l'integrazione personalizzata di Lambda. Per un endpoint HTTP, puoi avere l'integrazione proxy di HTTP o l'integrazione personalizzata di HTTP. Per un'azione AWS di servizio, è disponibile l' AWS integrazione solo del tipo non proxy. API Gateway supporta anche l'integrazione fittizia, in cui API Gateway serve come endpoint di integrazione per rispondere a una richiesta del metodo.

L'integrazione personalizzata Lambda è un caso speciale di integrazione, in cui AWS l'endpoint di integrazione corrisponde all'azione di richiamo della funzione del servizio Lambda.

A livello di programmazione, è possibile scegliere un tipo di integrazione configurando la proprietà `type` della risorsa `Integration`. Per l'integrazione proxy Lambda, il valore è `AWS_PROXY`. Per l'integrazione personalizzata Lambda e per tutte le altre integrazioni AWS , è `AWS`. Per l'integrazione proxy HTTP e l'integrazione HTTP, il valore è, rispettivamente, `HTTP_PROXY` e `HTTP`. Per l'integrazione fittizia, il valore `type` è `MOCK`.

L'integrazione proxy Lambda supporta una configurazione di integrazione semplificata con una singola funzione Lambda. La configurazione è semplice e può evolvere con il back-end senza annullare le impostazioni esistenti. Per questi motivi, è altamente consigliata per l'integrazione con una funzione Lambda.

Al contrario, l'integrazione personalizzata Lambda consente il riutilizzo di modelli di mappatura configurati per vari endpoint di integrazione che presentano requisiti simili in termini di formato di dati

di input e di output. La configurazione è più complessa ed è consigliata per scenari applicativi più avanzati.

Analogamente, l'integrazione proxy HTTP ha una configurazione di integrazione più semplice e può evolvere con il back-end senza annullare le impostazioni esistenti. L'integrazione personalizzata HTTP è più complessa da configurare, ma consente il riutilizzo di modelli di mappatura configurati per altri endpoint di integrazione.

L'elenco che segue riepiloga i tipi di integrazione supportati:

- `AWS`: questo tipo di integrazione consente a un'API di esporre le operazioni dei servizi AWS . Nell'integrazione `AWS`, devi configurare la richiesta e la risposta di integrazione, oltre a impostare le mappature di dati necessarie dalla richiesta di metodo alla richiesta di integrazione e dalla risposta di integrazione alla risposta di metodo.
- `AWS_PROXY`: questo tipo di integrazione consente l'integrazione di un metodo API con un'operazione di chiamata della funzione Lambda tramite una configurazione di integrazione semplificata, flessibile e versatile. Questa integrazione si basa su interazioni dirette tra il client e la funzione Lambda integrata.

  Con questo tipo di integrazione, noto anche come integrazione proxy Lambda, non devi configurare la richiesta o la risposta di integrazione. API Gateway trasferisce la richiesta in entrata dal client come input alla funzione Lambda back-end. La funzione Lambda integrata prende l'input di questo formato e analizza l'input di tutte le fonti disponibili, incluse le intestazioni delle richieste, le variabili di percorso dell'URL, i parametri delle stringhe di query e il corpo applicabile. La funzione restituisce il risultato seguendo questo formato di output.

  Questo è il tipo di integrazione preferito per chiamare una funzione Lambda tramite API Gateway e non è applicabile ad altre azioni di AWS servizio, incluse le azioni Lambda diverse dall'azione di richiamo della funzione.
- `HTTP`: questo tipo di integrazione consente a un'API di esporre gli endpoint HTTP nel back-end. Con l'integrazione `HTTP`, nota anche come integrazione HTTP personalizzata, devi configurare l richiesta e la risposta di integrazione. Devi configurare le mappature di dati necessarie dalla richiesta di metodo alla richiesta di integrazione e dalla risposta di integrazione alla risposta di metodo.
- `HTTP_PROXY`: l'integrazione proxy HTTP consente a un cliente di accedere agli endpoint HTTP di back-end con una configurazione di integrazione semplificata e un singolo metodo API. Non devi configurare la richiesta o la risposta di integrazione. API Gateway trasferisce la richiesta in entrata dal client all'endpoint HTTP e la risposta in uscita dall'endpoint HTTP al client.

- `MOCK`: questo tipo di integrazione consente ad API Gateway di restituire una risposta senza inviare la richiesta fino al back-end. Si tratta di un'opzione utile per il test di API, poiché può essere utilizzata per testare la configurazione di integrazione senza sostenere costi per l'utilizzo del back-end e per favorire uno sviluppo collaborativo di un'API.

  In uno sviluppo collaborativo, un team può isolare la propria attività di sviluppo impostando simulazioni di componenti API di proprietà di altri team utilizzando le integrazioni `MOCK`. Questa opzione viene anche utilizzata per restituire intestazioni relative alla configurazione CORS per verificare che il metodo API ne consenta l'accesso. Di fatto, la console API Gateway integra il metodo `OPTIONS` per supportare la configurazione CORS con un'integrazione fittizia. Le risposte del gateway sono altri esempi di integrazioni fittizie.

Configurazione di un'integrazione proxy mediante una risorsa proxy

Per configurare un'integrazione proxy in un'API di API Gateway mediante una risorsa proxy, completa le attività seguenti:

- Crea una risorsa proxy con una variabile di percorso greedy `{proxy+}`.
- Imposta il metodo `ANY` sulla risorsa proxy.
- Integra la risorsa e il metodo con un back-end utilizzando il tipo di integrazione HTTP o Lambda.

> **Note**
>
> Anche se normalmente vengono utilizzati insieme, i metodi `ANY`, i tipi di integrazione proxy e le variabili di percorso greedy sono caratteristiche indipendenti. È possibile configurare un metodo HTTP specifico per una risorsa greedy o applicare tipi di integrazione non proxy a una risorsa proxy.

In API Gateway la gestione dei metodi prevede restrizioni e limitazioni specifiche per entrambi tipi di integrazione proxy, Lambda e HTTP. Per informazioni dettagliate, vedi the section called “Note importanti”.

> **Note**
>
> Quando si utilizza l'integrazione proxy con un passthrough, API Gateway restituisce l'intestazione `Content-Type:application/json` di default, se non è specificato il tipo di contenuto di un payload.

Una risorsa proxy è più efficace se integrata con un back-end tramite l'integrazione proxy HTTP o l'integrazione proxy Lambda.

Integrazione proxy HTTP mediante una risorsa proxy

L'integrazione proxy HTTP, designata da `HTTP_PROXY` nell'API REST API Gateway, consente di integrare una richiesta del metodo con un endpoint HTTP di back-end. Con questo tipo di integrazione, API Gateway si limita a trasferire richiesta e risposta complete tra il front-end e il back-end, seguendo restrizioni e limitazioni specifiche.

> **Note**
>
> L'integrazione proxy HTTP supporta le intestazioni multi-valore e le stringhe di query.

Quando si applica l'integrazione proxy HTTP a una risorsa proxy, è possibile configurare l'API in modo che esponga una parte di un'intera gerarchia di endpoint del back-end HTTP con un'unica configurazione di integrazione. Supponi ad esempio che il back-end del sito Web sia organizzato in più rami dei nodi di struttura del nodo radice (`/site`) come: $/site/a_0/a_1/.../a_N$, $/site/b_0/b_1/.../b_M$ e così via. Se integri il metodo `ANY` di una risorsa proxy `/api/{proxy+}` con gli endpoint del back-end con percorsi URL `/site/{proxy}`, una singola richiesta di integrazione può supportare qualsiasi operazione HTTP (GET, POST e così via) ovunque in $[a_0, a_1, ..., a_N, b_0, b_1, ...b_M, ...]$. Se invece applichi un'integrazione proxy a un metodo HTTP specifico, ad esempio `GET`, la richiesta di integrazione risultante supporta le operazioni specificate (`GET`) su qualsiasi nodo di back-end.

Integrazione proxy Lambda mediante una risorsa proxy

L'integrazione proxy Lambda, designata da `AWS_PROXY` nell'API REST API Gateway, consente di integrare una richiesta del metodo con una funzione Lambda nel back-end. Con questo tipo di integrazione, API Gateway applica un modello di mappatura predefinito per inviare l'intera richiesta alla funzione Lambda e trasforma l'output di tale funzione in risposte HTTP.

Analogamente, puoi applicare l'integrazione proxy Lambda a una risorsa proxy di `/api/{proxy+}` per configurare una sola integrazione e fare in modo che una funzione Lambda back-end reagisca individualmente alle modifiche apportate in una delle risorse API in `/api`.

Configurazione di una richiesta di integrazione API tramite la console API Gateway

La configurazione di un metodo API definisce il metodo e descrive i suoi comportamenti. Per configurare un metodo, è necessario specificare una risorsa su cui il metodo è esposto, inclusa la root ("/"), un metodo HTTP (`GET`, `POST` e così via) e il modo in cui sarà integrato con il relativo back-end. La richiesta e la risposta di metodo specificano il contratto con l'app di chiamata e stipulano i parametri che l'API può ricevere e l'aspetto della risposta.

Le seguenti procedure descrivono come utilizzare la console API Gateway per creare una richiesta di integrazione.

Argomenti

- Configura un'integrazione Lambda
- Configura un'integrazione HTTP
- Configura un'integrazione AWS del servizio
- Configura un'integrazione fittizia

Configura un'integrazione Lambda

Usa un'integrazione di funzioni Lambda per integrare la tua API con una funzione Lambda. A livello di API, sarà un tipo di integrazione `AWS` se si crea un'integrazione non proxy o un tipo di integrazione `AWS_PROXY` se si crea un'integrazione proxy.

Per configurare un'integrazione Lambda

1. Nel riquadro Risorse scegli Crea metodo.
2. Per Tipo di metodo seleziona un metodo HTTP.
3. Per Integration type (Tipo di integrazione), scegli Lambda Function (Funzione Lambda).
4. Per utilizzare un'integrazione proxy Lambda, attiva Integrazione proxy Lambda. Per ulteriori informazioni sulle integrazioni proxy Lambda, consulta the section called “ Informazioni sull'integrazione proxy Lambda ”.
5. Per Funzione Lambda immetti il nome della funzione Lambda.

Se utilizzi una funzione Lambda in una regione diversa da quella dell'API, seleziona la regione dal menu a discesa e inserisci il nome della funzione Lambda. Se usi una funzione Lambda tra diversi account immetti il nome della risorsa Amazon (ARN) della funzione.

6. Per utilizzare il valore di timeout predefinito di 29 secondi, mantieni attiva l'opzione Timeout predefinito. Per impostare un timeout personalizzato, scegli Timeout predefinito e immetti un valore di timeout compreso tra `50` e `29000` millisecondi.
7. (Facoltativo) È possibile configurare le impostazioni di richiesta del metodo utilizzando i seguenti menu a discesa. Scegli le impostazioni della richiesta del metodo e configura la tua richiesta del metodo. Per ulteriori informazioni, consulta il passaggio 3 dithe section called “Modifica una richiesta di metodo nella console”.

   È inoltre possibile configurare le impostazioni di richiesta del metodo dopo aver creato il metodo.
8. Scegli Crea metodo.

Configura un'integrazione HTTP

Utilizza un'integrazione HTTP per integrare la tua API con un endpoint HTTP. A livello di API, questa è un'integrazione di tipo `HTTP`.

Per configurare un'integrazione HTTP

1. Nel riquadro Risorse scegli Crea metodo.
2. Per Tipo di metodo seleziona un metodo HTTP.
3. Per il tipo di integrazione, scegli HTTP.
4. Per utilizzare un'integrazione proxy HTTP, attiva Integrazione proxy HTTP. Per ulteriori informazioni sulle integrazioni proxy HTTP, consulta the section called “Configurazione di integrazioni proxy HTTP in API Gateway”.
5. Per HTTP method (Metodo HTTP) scegliere il tipo di metodo HTTP che corrisponde maggiormente al metodo nel back-end HTTP.
6. Per URL endpoint immetti l'URL del back-end HTTP che desideri venga utilizzato dal metodo.
7. Per Gestione contenuti seleziona un comportamento di gestione dei contenuti.
8. Per utilizzare il valore di timeout predefinito di 29 secondi, mantieni attiva l'opzione Timeout predefinito. Per impostare un timeout personalizzato, scegli Timeout predefinito e immetti un valore di timeout compreso tra `50` e `29000` millisecondi.

9. (Facoltativo) È possibile configurare le impostazioni di richiesta del metodo utilizzando i seguenti menu a discesa. Scegli le impostazioni della richiesta del metodo e configura la tua richiesta del metodo. Per ulteriori informazioni, consulta il passaggio 3 dithe section called “Modifica una richiesta di metodo nella console”.

   È inoltre possibile configurare le impostazioni di richiesta del metodo dopo aver creato il metodo.
10. Scegli Crea metodo.

Configura un'integrazione AWS del servizio

Utilizza un'integrazione AWS di servizi per integrare la tua API direttamente con un AWS servizio. A livello di API, questa è un'integrazione di tipo `AWS`.

Per configurare un'API Gateway API procedi come segue:

- Crea una nuova funzione Lambda.
- Imposta l'autorizzazione per le risorse alla funzione Lambda.
- Esegui qualsiasi altra azione del servizio Lambda.

È necessario scegliere Servizio AWS .

Per configurare un'integrazione AWS di servizi

1. Nel riquadro Risorse scegli Crea metodo.
2. Per Tipo di metodo seleziona un metodo HTTP.
3. Per Tipo di integrazione, scegli AWS servizio.
4. Per AWS Regione, scegli la AWS regione che desideri utilizzare questo metodo per avviare l'azione.
5. Per l'AWS assistenza, scegli il AWS servizio che desideri chiamare con questo metodo.
6. Per AWS sottodominio, inserisci il sottodominio utilizzato dal AWS servizio. In genere, questo spazio rimane vuoto. Alcuni servizi AWS supportano i sottodomini come parte degli host. Consulta la documentazione relativa al servizio per la disponibilità e, se disponibili, ulteriori dettagli.
7. In HTTP method (Metodo HTTP) scegliere il tipo di metodo HTTP che corrisponde all'operazione. Per il tipo di metodo HTTP, consulta la documentazione di riferimento dell'API per il AWS servizio che hai scelto per AWS il servizio.

8. Per Tipo di operazione seleziona Usa nome operazione per utilizzare un'operazione API o Utilizza sostituzione percorso per usare un percorso personalizzato della risorsa. Per le azioni disponibili e i percorsi di risorse personalizzati, consulta la documentazione di riferimento dell'API per il AWS servizio che hai scelto per il AWS servizio.
9. Inserisci il Nome azione o la Sostituzione percorso.
10. In Ruolo di esecuzione immetti il nome della risorsa Amazon (ARN) del ruolo IAM utilizzato dal metodo per chiamare l'operazione.

    Per creare il ruolo IAM puoi adattare le istruzioni disponibili in the section called “Fase 1: creare il ruolo di esecuzione del proxy del AWS servizio”. Specifica una policy di accesso predefinita con il seguente formato e il numero desiderato di dichiarazioni relativo a risorse e operazioni:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "action-statement"
      ],
      "Resource": [
        "resource-statement"
      ]
    },
    ...
  ]
}
```

    Per la sintassi dell'istruzione relativa alle azioni e alle risorse, consulta la documentazione del AWS servizio che hai scelto per il AWS servizio.

    Per la relazione di fiducia del ruolo IAM, specificare quanto segue, in modo da consentire ad API Gateway di intervenire per conto dell'account AWS :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
```

```
      "Principal": {
        "Service": "apigateway.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

11. Per utilizzare il valore di timeout predefinito di 29 secondi, mantieni attiva l'opzione Timeout predefinito. Per impostare un timeout personalizzato, scegli Timeout predefinito e immetti un valore di timeout compreso tra `50` e `29000` millisecondi.
12. (Facoltativo) È possibile configurare le impostazioni di richiesta del metodo utilizzando i seguenti menu a discesa. Scegli le impostazioni della richiesta del metodo e configura la tua richiesta del metodo. Per ulteriori informazioni, consulta il passaggio 3 dithe section called “Modifica una richiesta di metodo nella console”.

    È inoltre possibile configurare le impostazioni di richiesta del metodo dopo aver creato il metodo.
13. Scegli Crea metodo.

Configura un'integrazione fittizia

Utilizza un'integrazione fittizia se desideri che API Gateway funga da backend per restituire risposte statiche. A livello di API, questa è un'integrazione di tipo `MOCK`. In genere, puoi utilizzare l'integrazione `MOCK` quando la tua API non è ancora quella definitiva, ma vuoi generare risposte API per liberare i team relativi e dedicarli al test. Per il metodo `OPTION`, API Gateway imposta l'integrazione `MOCK` come predefinita per restituire intestazioni abilitate a CORS per la risorsa API applicata.

Per configurare un'integrazione fittizia

1. Nel riquadro Risorse scegli Crea metodo.
2. Per Tipo di metodo seleziona un metodo HTTP.
3. Per il tipo di integrazione, scegli Mock.
4. (Facoltativo) È possibile configurare le impostazioni di richiesta del metodo utilizzando i seguenti menu a discesa. Scegli le impostazioni della richiesta del metodo e configura la tua richiesta del metodo. Per ulteriori informazioni, consulta il passaggio 3 dithe section called “Modifica una richiesta di metodo nella console”.

    È inoltre possibile configurare le impostazioni di richiesta del metodo dopo aver creato il metodo.

5. Scegli Crea metodo.

Configurazione di una risposta di integrazione in API Gateway

Per un'integrazione non-proxy, devi impostare almeno una risposta di integrazione come predefinita, per trasmettere il risultato restituito dal back-end al client. Puoi scegliere di trasmettere il risultato invariato o di trasformare i dati della risposta di integrazione in dati della risposta di metodo se i formati sono diversi.

Per un'integrazione proxy, API Gateway passa automaticamente l'output del back-end al client come risposta HTTP. Non devi configurare la risposta del metodo o la risposta di integrazione.

Per configurare una risposta di integrazione, è necessario completare le seguenti attività obbligatorie e facoltative:

1. Specifica un codice di stato HTTP di una risposta di metodo su cui sono mappati i dati della risposta di integrazione. Questo dato è obbligatorio.
2. Definisci un'espressione regolare per scegliere che l'output del back-end sia rappresentato da questa risposta di integrazione. Se questo spazio rimane vuoto, la risposta corrisponde a quella predefinita utilizzata per cogliere qualsiasi risposta non ancora configurata.
3. Se necessario, dichiara le mappature che consistono di coppie chiave-valore per mappare parametri di risposta di integrazione specifici su parametri di risposta di metodo.
4. Se necessario, aggiungi modelli di mappatura del corpo per trasformare i payload di risposte di integrazione specifiche in payload di risposta di metodo.
5. Se necessario, specifica come gestire una conversione dei tipi per un payload binario.

Una risposta di integrazione è una risposta HTTP contenente la risposta del back-end. Per un endpoint HTTP, la risposta del back-end è una risposta HTTP. Il codice di stato della risposta di integrazione può riferirsi al codice di stato restituito dal back-end, mentre il corpo della risposta di integrazione coincide con il payload restituito dal back-end. Per un endpoint Lambda, la risposta del back-end è l'output restituito dalla funzione Lambda. Con l'integrazione Lambda, l'output della funzione Lambda viene restituito come risposta `200 OK`. Il payload può contenere il risultato come dati JSON, incluso una stringa o un oggetto JSON, o un messaggio di errore come oggetto JSON. Puoi assegnare un'espressione regolare alla proprietà selectionPattern per mappare una risposta di errore a una risposta di errore HTTP adeguata. Per ulteriori informazioni sulle risposte di errore della funzione Lambda, consulta Gestione degli errori Lambda in API Gateway. Con l'integrazione proxy di Lambda, la funzione Lambda deve restituire l'output nel formato seguente:

```
{
    statusCode: "...",            // a valid HTTP status code
    headers: {
        custom-header: "..."      // any API-specific custom header
    },
    body: "...",                  // a JSON string.
    isBase64Encoded:  true|false  // for binary support
}
```

Non è necessario mappare la risposta della funzione Lambda alla risposta HTTP appropriata.

Per restituire il risultato al client, configura la risposta di integrazione per trasmettere la risposta invariata dell'endpoint alla risposta di metodo corrispondente. Oppure puoi mappare i dati di risposta dell'endpoint ai dati di risposta del metodo. Tra i dati di risposta che possono essere mappati figurano il codice di stato della risposta, i parametri di intestazione della risposta e il corpo della risposta. Se per il codice di stato restituito non è stata definita una risposta del metodo, API Gateway restituisce l'errore 500. Per ulteriori informazioni, consulta Utilizza un modello di mappatura per sovrascrivere i parametri API di richiesta e risposta e i codici di stato di un.

Integrazioni Lambda per in Gateway REST APIs API

È possibile integrare un API metodo con una funzione Lambda utilizzando l'integrazione proxy Lambda o l'integrazione non proxy Lambda (personalizzata).

Nell'integrazione proxy Lambda la configurazione richiesta è semplice. Imposta il HTTP metodo di integrazione suPOST, l'endpoint URI di integrazione sull'azione di invocazione ARN della funzione Lambda di una specifica funzione Lambda e concedi a API Gateway l'autorizzazione a chiamare la funzione Lambda per tuo conto.

Nell'integrazione non proxy Lambda, oltre alla procedura di configurazione dell'integrazione proxy, devi specificare anche il modo in cui i dati della richiesta in entrata vengono mappati alla richiesta di integrazione e il modo in cui i dati della risposta di integrazione risultante vengono mappati alla risposta del metodo.

Argomenti

- Integrazioni proxy Lambda in Gateway API
- Configura integrazioni personalizzate Lambda in Gateway API
- Configurazione della chiamata asincrona della funzione Lambda back-end
- Gestione degli errori Lambda in API Gateway

Integrazioni proxy Lambda in Gateway API

La sezione seguente mostra come utilizzare un'integrazione proxy Lambda.

Argomenti

- Comprendi l'integrazione del API proxy Gateway Lambda
- Supporto per le intestazioni multi-valore e i parametri di stringa di query
- Formato di input di una funzione Lambda per l'integrazione proxy
- Formato di output di una funzione Lambda per l'integrazione proxy
- Configurare l'integrazione del proxy Lambda per API Gateway utilizzando AWS CLI
- Configurare una risorsa proxy con integrazione del proxy Lambda con una definizione aperta API

Comprendi l'integrazione del API proxy Gateway Lambda

L'integrazione del proxy Amazon API Gateway Lambda è un meccanismo semplice, potente e agile per creare un proxy API con una configurazione di un unico metodo. API Permette al client di chiamare una singola funzione Lambda nel back-end. La funzione accede a molte risorse o funzionalità di altri AWS servizi, inclusa la chiamata ad altre funzioni Lambda.

Nell'integrazione con proxy Lambda, quando un client invia una API richiesta, API Gateway passa alla funzione Lambda integrata un oggetto evento, tranne per il fatto che l'ordine dei parametri della richiesta non viene mantenuto. Questi dati di richiesta includono le intestazioni della richiesta, i parametri della stringa di query, le variabili di URL percorso, il payload e i dati di configurazione. API I dati di configurazione possono includere il nome della fase di distribuzione corrente, variabili di fasi, identità utente o contesto di autorizzazione (se presente). La funzione Lambda back-end analizza i dati delle richieste in entrata per stabilire la risposta da restituire. Affinché API Gateway passi l'output Lambda come API risposta al client, la funzione Lambda deve restituire il risultato in questo formato.

Poiché API Gateway non interviene molto tra il client e la funzione Lambda di backend per l'integrazione del proxy Lambda, il client e la funzione Lambda integrata possono adattarsi alle modifiche reciproche senza interrompere la configurazione di integrazione esistente di. API Per consentirlo, il client deve seguire i protocolli di applicazione attuati dalla funzione Lambda back-end.

Puoi configurare un'integrazione proxy Lambda per qualsiasi API metodo. Tuttavia, l'integrazione del proxy Lambda è più potente quando è configurata per un API metodo che coinvolge una risorsa proxy generica. La risorsa proxy generica può essere indicata dalla speciale variabile di percorso

preconfigurata `{proxy+}`, dal segnaposto del metodo catch-all `ANY` o da entrambi. Il client può passare l'input alla funzione Lambda back-end nella richiesta in entrata come parametri di richiesta o payload applicabile. I parametri di richiesta includono intestazioni, variabili di URL percorso, parametri della stringa di query e il payload applicabile. La funzione Lambda integrata verifica tutte le sorgenti di input prima di elaborare la richiesta e rispondere al client con messaggi di errore significativi, se manca qualcuno degli input richiesti.

Quando si chiama un API metodo integrato con il HTTP metodo generico di `ANY` e la risorsa generica di`{proxy+}`, il client invia una richiesta con un HTTP metodo particolare al posto di. `ANY` Il client specifica inoltre un URL percorso particolare anziché e include tutte le intestazioni richieste`{proxy+}`, i parametri della stringa di query o un payload applicabile.

L'elenco seguente riassume i comportamenti di runtime dei diversi API metodi con l'integrazione del proxy Lambda:

- `ANY /{proxy+}`: il client deve scegliere un HTTP metodo particolare, deve impostare una particolare gerarchia di percorsi di risorse e può impostare qualsiasi intestazione, parametro della stringa di query e payload applicabile per passare i dati come input alla funzione Lambda integrata.
- `ANY /res`: il client deve scegliere un HTTP metodo particolare e può impostare qualsiasi intestazione, parametro della stringa di query e payload applicabile per passare i dati come input alla funzione Lambda integrata.
- `GET|POST|PUT|... /{proxy+}`: per passare i dati come input alla funzione Lambda integrata, il client può impostare una specifica gerarchia di percorso di risorsa, qualsiasi intestazione, parametro di stringa di query e payload applicabile.
- `GET|POST|PUT|... /res/{path}/...`: per passare i dati di input alla funzione Lambda integrata, il client deve scegliere un segmento di percorso specifico (per la variabile `{path}`) e può impostare qualsiasi intestazione di richiesta, parametro di stringa di query e payload applicabile.
- `GET|POST|PUT|... /res`: per passare i dati di input alla funzione Lambda integrata, il client può scegliere qualsiasi intestazione di richiesta, parametro di stringa di query e payload applicabile.

La risorsa proxy `{proxy+}` e quella personalizzata `{custom}` sono entrambe espresse come variabili di percorso preconfigurate. Tuttavia `{proxy+}` può fare riferimento a qualsiasi risorsa in una gerarchia di percorso, mentre `{custom}` fa riferimento solo a un segmento di percorso specifico. Ad esempio, un negozio di generi alimentari potrebbe organizzare il proprio inventario di prodotti online in base ai nomi dei reparti, alle categorie di produzione e ai tipi di prodotto. Il sito Web del negozio potrebbe quindi rappresentare i prodotti disponibili in base alle seguenti variabili di percorso

preconfigurate di risorse personalizzate: `/{department}/{produce-category}/{product-type}`. Ad esempio le mele sono rappresentate da `/produce/fruit/apple` e le carote da `/produce/vegetables/carrot`. Può anche usare `/{proxy+}` per rappresentare qualsiasi reparto, categoria di produzione o tipo di prodotto che un cliente può cercare quando fa acquisti nel negozio online. Ad esempio, `/{proxy+}` può fare riferimento a uno qualsiasi degli articoli seguenti:

- `/produce`
- `/produce/fruit`
- `/produce/vegetables/carrot`

Per permettere ai clienti di cercare i prodotti disponibili, la categoria di produzione e il reparto del negozio associato, puoi esporre un singolo metodo `GET /{proxy+}` con autorizzazioni di sola lettura. Analogamente, per consentire a un superiore di aggiornare l'inventario del reparto `produce`, puoi impostare un altro metodo singolo `PUT /produce/{proxy+}` con autorizzazioni di lettura/scrittura. Per permettere a un cassiere di aggiornare la quantità complessiva di un tipo di ortaggio, puoi configurare un metodo `POST /produce/vegetables/{proxy+}` con autorizzazioni di lettura/scrittura. Per permettere a un responsabile di negozio di eseguire tutte le operazioni possibili per tutti i prodotti disponibili, lo sviluppatore di negozi online può esporre il metodo `ANY /{proxy+}` con autorizzazioni di lettura/scrittura. In ogni caso, al runtime il cliente o il dipendente deve selezionare un prodotto specifico di un determinato tipo in un reparto scelto, una particolare categoria di produzione in un reparto scelto o un reparto specifico.

Per ulteriori informazioni sulla configurazione delle integrazioni proxy API Gateway, consulta. Configurazione di un'integrazione proxy mediante una risorsa proxy

Per l'integrazione proxy è necessario che il client abbia una conoscenza più dettagliata dei requisiti del back-end. Pertanto, per garantire prestazioni dell'app e un'esperienza utente ottimali, lo sviluppatore di back-end deve comunicare chiaramente allo sviluppatore del client i requisiti del back-end e fornire un sistema solido di notifica degli errori quando i requisiti non vengono soddisfatti.

Supporto per le intestazioni multi-valore e i parametri di stringa di query

APIGateway supporta più intestazioni e parametri di stringhe di query con lo stesso nome. Le intestazioni multi-valore, quelle con valore singolo e le intestazioni di valore e parametri possono essere combinati nelle stesse richieste e risposte. Per ulteriori informazioni, consulta Formato di input di una funzione Lambda per l'integrazione proxy e Formato di output di una funzione Lambda per l'integrazione proxy.

Formato di input di una funzione Lambda per l'integrazione proxy

Nell'integrazione del proxy Lambda, API Gateway mappa l'intera richiesta del client al `event` parametro di input della funzione Lambda di backend. L'esempio seguente mostra la struttura di un evento che API Gateway invia a un'integrazione proxy Lambda.

```
{
  "resource": "/my/path",
  "path": "/my/path",
  "httpMethod": "GET",
  "headers": {
    "header1": "value1",
    "header2": "value1,value2"
  },
  "multiValueHeaders": {
    "header1": [
      "value1"
    ],
    "header2": [
      "value1",
      "value2"
    ]
  },
  "queryStringParameters": {
    "parameter1": "value1,value2",
    "parameter2": "value"
  },
  "multiValueQueryStringParameters": {
    "parameter1": [
      "value1",
      "value2"
    ],
    "parameter2": [
      "value"
    ]
  },
  "requestContext": {
    "accountId": "123456789012",
    "apiId": "id",
    "authorizer": {
      "claims": null,
      "scopes": null
    },
```

```
    "domainName": "id.execute-api.us-east-1.amazonaws.com",
    "domainPrefix": "id",
    "extendedRequestId": "request-id",
    "httpMethod": "GET",
    "identity": {
      "accessKey": null,
      "accountId": null,
      "caller": null,
      "cognitoAuthenticationProvider": null,
      "cognitoAuthenticationType": null,
      "cognitoIdentityId": null,
      "cognitoIdentityPoolId": null,
      "principalOrgId": null,
      "sourceIp": "IP",
      "user": null,
      "userAgent": "user-agent",
      "userArn": null,
      "clientCert": {
        "clientCertPem": "CERT_CONTENT",
        "subjectDN": "www.example.com",
        "issuerDN": "Example issuer",
        "serialNumber": "a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1",
        "validity": {
          "notBefore": "May 28 12:30:02 2019 GMT",
          "notAfter": "Aug  5 09:36:04 2021 GMT"
        }
      }
    },
    "path": "/my/path",
    "protocol": "HTTP/1.1",
    "requestId": "id=",
    "requestTime": "04/Mar/2020:19:15:17 +0000",
    "requestTimeEpoch": 1583349317135,
    "resourceId": null,
    "resourcePath": "/my/path",
    "stage": "$default"
  },
  "pathParameters": null,
  "stageVariables": null,
  "body": "Hello from Lambda!",
  "isBase64Encoded": false
}
```

> **Note**
>
> Nell'input:
>
> - La chiave `headers` può contenere solo una decina di intestazioni di valore.
> - La chiave `multiValueHeaders` può contenere intestazioni multi-valore e il valore di una decina di intestazioni.
> - Se si specificano valori per entrambi `headers` e`multiValueHeaders`, API Gateway li unisce in un unico elenco. Se la stessa coppia chiave-valore viene specificata in entrambi, solo i valori di `multiValueHeaders` verranno visualizzati nell'elenco risultante.

Nell'input per la funzione Lambda back-end l'oggetto `requestContext` è una mappa di coppie chiave/valore. In ogni coppia la chiave è il nome di una proprietà della variabile $context e il valore è il valore della proprietà. APIGateway potrebbe aggiungere nuove chiavi alla mappa.

A seconda delle funzionalità abilitate, la `requestContext` mappa può variare da API aAPI. Ad esempio, nell'esempio precedente non è specificato alcun tipo di autorizzazione e di conseguenza non è presente alcuna proprietà `$context.authorizer.*` o `$context.identity.*`. Quando viene specificato un tipo di autorizzazione, API Gateway trasmette le informazioni degli utenti autorizzati all'endpoint di integrazione in un `requestContext.identity` oggetto nel modo seguente:

- Quando il tipo di autorizzazione è `AWS_IAM`, le informazioni sugli utenti autorizzati includono proprietà `$context.identity.*`.
- Quando il tipo di autorizzazione è `COGNITO_USER_POOLS` (autorizzazione di Amazon Cognito), le informazioni sugli utenti autorizzati includono le proprietà `$context.identity.cognito*` e `$context.authorizer.claims.*`.
- Quando il tipo di autorizzazione è `CUSTOM` (autorizzazione Lambda), le informazioni sugli utenti autorizzati includono `$context.authorizer.principalId` e altre proprietà `$context.authorizer.*` applicabili.

Formato di output di una funzione Lambda per l'integrazione proxy

Nell'integrazione del proxy Lambda, API Gateway richiede la funzione Lambda di backend per restituire l'output secondo il seguente formato: JSON

```
{
    "isBase64Encoded": true|false,
    "statusCode": httpStatusCode,
    "headers": { "headerName": "headerValue", ... },
    "multiValueHeaders": { "headerName": ["headerValue", "headerValue2", ...], ... },
    "body": "..."
}
```

Nell'output:

- Le chiavi `headers` e `multiValueHeaders` possono non essere specificate se non devono essere restituite intestazioni di risposta aggiuntive.
- La chiave `headers` può contenere solo una decina di intestazioni di valore.
- La chiave `multiValueHeaders` può contenere intestazioni multi-valore e il valore di una decina di intestazioni. È possibile utilizzare la chiave `multiValueHeaders` per specificare tutte le intestazioni aggiuntive, incluse quelle con valore singolo.
- Se si specificano valori per entrambi `headers` e`multiValueHeaders`, API Gateway li unisce in un unico elenco. Se la stessa coppia chiave-valore viene specificata in entrambi, solo i valori di `multiValueHeaders` verranno visualizzati nell'elenco risultante.

CORSPer abilitare l'integrazione del proxy Lambda, è necessario aggiungere `Access-Control-Allow-Origin:`*`domain-name`* all'output. `headers` *`domain-name`*può essere `*` per qualsiasi nome di dominio. Viene eseguito il marshalling del parametro `body` di output al front-end come payload di risposta del metodo. Se `body` è un BLOB binario, puoi codificarlo come stringa con codifica Base64 impostando `isBase64Encoded` su `true` e configurando `*/*` come Binary Media Type (Tipo multimediale binario). In caso contrario, puoi impostarlo su `false` o lasciarlo non specificato.

> **Note**
>
> Per ulteriori informazioni sull'abilitazione del supporto binario, consulta Abilitazione del supporto binario tramite la console Gateway API. Per un esempio di funzione Lambda, consulta Restituisci supporti binari da un'integrazione proxy Lambda in Gateway API.

Se l'output della funzione ha un formato diverso, API Gateway restituisce una risposta `502 Bad Gateway` di errore.

Per restituire una risposta in una funzione Lambda in Node.js, è possibile utilizzare comandi come i seguenti:

- Per ottenere un buon risultato, effettuare la chiamata a `callback(null, {"statusCode": 200, "body": "results"})`.
- Per generare un'eccezione, chiama `callback(new Error('internal server error'))`.
- Per un errore lato client (se, ad esempio, un parametro obbligatorio è mancante), puoi chiamare `callback(null, {"statusCode": 400, "body": "Missing parameters of ..."})` per restituire l'errore senza generare un'eccezione.

In una funzione Lambda `async` in Node.js, la sintassi equivalente è la seguente:

- Per ottenere un buon risultato, effettuare la chiamata a `return {"statusCode": 200, "body": "results"}`.
- Per generare un'eccezione, chiama `throw new Error("internal server error")`.
- Per un errore lato client (se, ad esempio, un parametro obbligatorio è mancante), puoi chiamare `return {"statusCode": 400, "body": "Missing parameters of ..."}` per restituire l'errore senza generare un'eccezione.

Configurare l'integrazione del proxy Lambda per API Gateway utilizzando AWS CLI

In questa sezione, mostriamo come configurare un'integrazione API con il proxy Lambda utilizzando il. AWS CLI Per istruzioni dettagliate sull'utilizzo della console API Gateway per configurare una risorsa proxy con l'integrazione del proxy Lambda, vedere. Tutorial: Creare un'integrazione REST API con un proxy Lambda

Ad esempio, utilizziamo la seguente funzione Lambda di esempio come backend di: API

```
export const handler = function(event, context, callback) {
    console.log('Received event:', JSON.stringify(event, null, 2));
    var res ={
        "statusCode": 200,
        "headers": {
            "Content-Type": "*/*"
        }
    };
    var greeter = 'World';
    if (event.greeter && event.greeter!==""") {
```

```
        greeter =  event.greeter;
    } else if (event.body && event.body !== "") {
        var body = JSON.parse(event.body);
        if (body.greeter && body.greeter !== "") {
            greeter = body.greeter;
        }
    } else if (event.queryStringParameters && event.queryStringParameters.greeter &&
 event.queryStringParameters.greeter !== "") {
        greeter = event.queryStringParameters.greeter;
    } else if (event.multiValueHeaders && event.multiValueHeaders.greeter &&
 event.multiValueHeaders.greeter != "") {
        greeter = event.multiValueHeaders.greeter.join(" and ");
    } else if (event.headers && event.headers.greeter && event.headers.greeter != "") {
        greeter = event.headers.greeter;
    }

    res.body = "Hello, " + greeter + "!";
    callback(null, res);
};
```

Rispetto alla configurazione di integrazione personalizzata Lambda inthe section called “ Configurazione di integrazioni personalizzate Lambda ”, l'input di questa funzione Lambda può essere espresso nei parametri e nel corpo della richiesta. Avrai maggiore libertà per permettere al client di passare gli stessi dati di input. Qui il client può passare il nome del mittente della formula di saluto come parametro della stringa di query, intestazione o proprietà del corpo. La funzione può anche supportare l'integrazione personalizzata Lambda. La API configurazione è più semplice. Non devi configurare la risposta del metodo o la risposta di integrazione.

Per configurare un'integrazione con proxy Lambda utilizzando AWS CLI

1. Chiama il `create-rest-api` comando per creare unAPI:

   ```
   aws apigateway create-rest-api --name 'HelloWorld (AWS CLI)' --region us-west-2
   ```

   Nota il `id` valore risultante API (`te6si5ach7`) nella risposta:

   ```
   {
       "name": "HelloWorldProxy (AWS CLI)",
       "id": "te6si5ach7",
       "createdDate": 1508461860
   }
   ```

Hai bisogno di API `id` tutta questa sezione.

2. Chiama il comando `get-resources` per ottenere il valore di `id` della risorsa root:

```
aws apigateway get-resources --rest-api-id te6si5ach7 --region us-west-2
```

La risposta di esito positivo sarà simile alla seguente:

```
{
    "items": [
        {
            "path": "/",
            "id": "krznpq9xpg"
        }
    ]
}
```

Prendi nota del valore di `id` (`krznpq9xpg`) della risorsa root. Ti servirà nella prossima fase e successivamente.

3. Chiama `create-resource` per creare una risorsa API gateway di`/greeting`:

```
aws apigateway create-resource --rest-api-id te6si5ach7 \
      --region us-west-2 \
      --parent-id krznpq9xpg \
      --path-part {proxy+}
```

La risposta con esito positivo è simile a quella riportata di seguito.

```
{
    "path": "/{proxy+}",
    "pathPart": "{proxy+}",
    "id": "2jf6xt",
    "parentId": "krznpq9xpg"
}
```

Prendi nota del valore `{proxy+}` della risorsa `id` (`2jf6xt`). Ti servirà per creare un metodo nella risorsa `/{proxy+}` nella prossima fase.

4. Chiama `put-method` per creare una richiesta del metodo `ANY` `ANY /{proxy+}`:

```
aws apigateway put-method --rest-api-id te6si5ach7 \
       --region us-west-2 \
       --resource-id 2jf6xt \
       --http-method ANY \
       --authorization-type "NONE"
```

La risposta con esito positivo è simile a quella riportata di seguito.

```
{
    "apiKeyRequired": false,
    "httpMethod": "ANY",
    "authorizationType": "NONE"
}
```

Questo API metodo consente al client di ricevere o inviare saluti dalla funzione Lambda nel backend.

5. Chiama `put-integration` per configurare l'integrazione del metodo `ANY /{proxy+}` con una funzione Lambda, denominata `HelloWorld`. Questa funzione risponde alla richiesta con un messaggio `"Hello, {name}!"` se è specificato il parametro `greeter`, altrimenti con un messaggio `"Hello, World!"` se il parametro della stringa di query non è impostato.

```
aws apigateway put-integration \
        --region us-west-2 \
        --rest-api-id te6si5ach7 \
        --resource-id 2jf6xt \
        --http-method ANY \
        --type AWS_PROXY \
        --integration-http-method POST \
        --uri arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:HelloWorld/invocations \
        --credentials arn:aws:iam::123456789012:role/apigAwsProxyRole
```

> **Important**
>
> Per le integrazioni Lambda, è necessario utilizzare il HTTP metodo di `POST` per la richiesta di integrazione, in base alla specifica dell'azione del servizio Lambda per le chiamate di funzioni. Il IAM ruolo delle politiche `apigAwsProxyRole` indispensabili che consentono al `apigateway` servizio di richiamare le funzioni Lambda. Per ulteriori

informazioni sulle IAM autorizzazioni, consulta. the section called “ APIModello di autorizzazioni Gateway per richiamare un API”

L'output corretto sarà simile al seguente:

```
{
    "passthroughBehavior": "WHEN_NO_MATCH",
    "cacheKeyParameters": [],
    "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:1234567890:function:HelloWorld/invocations",
    "httpMethod": "POST",
    "cacheNamespace": "vvom7n",
    "credentials": "arn:aws:iam::1234567890:role/apigAwsProxyRole",
    "type": "AWS_PROXY"
}
```

Invece di fornire un IAM ruolo per`credentials`, puoi chiamare il comando add-permission per aggiungere autorizzazioni basate sulle risorse. Questo è ciò che fa la console Gateway. API

6. Chiama `create-deployment` per API implementarlo su una `test` fase:

```
aws apigateway create-deployment --rest-api-id te6si5ach7 --stage-name test --
region us-west-2
```

7. Prova l'APIutilizzo dei seguenti URL comandi c in un terminale.

   Chiamando il API con il parametro della stringa di query di`?greeter=jane`:

```
curl -X GET 'https://te6si5ach7.execute-api.us-west-2.amazonaws.com/test/greeting?
greeter=jane'
```

   Chiamando il API con un parametro di `greeter:jane` intestazione pari a:

```
curl -X GET https://te6si5ach7.execute-api.us-west-2.amazonaws.com/test/hi \
  -H 'content-type: application/json' \
  -H 'greeter: jane'
```

   Chiamando il API con un corpo di`{"greeter":"jane"}`:

```
curl -X POST https://te6si5ach7.execute-api.us-west-2.amazonaws.com/test/hi \
  -H 'content-type: application/json' \
  -d '{ "greeter": "jane" }'
```

In tutti i casi, l'output è una risposta 200 con il corpo seguente:

```
Hello, jane!
```

Configurare una risorsa proxy con integrazione del proxy Lambda con una definizione aperta API

Per configurare una risorsa proxy con il tipo di integrazione proxy Lambda, crea una API risorsa con un parametro greedy path (ad esempio`/parent/{proxy+}`) e integra questa risorsa con un backend della funzione Lambda (ad esempio) sul metodo. `arn:aws:lambda:us-west-2:123456789012:function:SimpleLambda4ProxyResource` ANY Il parametro greedy path deve trovarsi alla fine del percorso della risorsa. API Analogamente a una risorsa non proxy, è possibile configurare la risorsa proxy utilizzando la console API Gateway, importando un file Open API definition o chiamando direttamente il API Gateway. REST API

Il seguente file di API API definizione aperta mostra un esempio di risorsa API con un proxy integrata con una funzione Lambda denominata. `SimpleLambda4ProxyResource`

OpenAPI 3.0

```
{
   "openapi": "3.0.0",
   "info": {
      "version": "2016-09-12T17:50:37Z",
      "title": "ProxyIntegrationWithLambda"
   },
   "paths": {
      "/{proxy+}": {
         "x-amazon-apigateway-any-method": {
            "parameters": [
               {
                  "name": "proxy",
                  "in": "path",
                  "required": true,
                  "schema": {
                     "type": "string"
```

```
                  }
               }
            ],
            "responses": {},
            "x-amazon-apigateway-integration": {
               "responses": {
                  "default": {
                     "statusCode": "200"
                  }
               },
               "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:123456789012:function:SimpleLambda4ProxyResource/
invocations",
               "passthroughBehavior": "when_no_match",
               "httpMethod": "POST",
               "cacheNamespace": "roq9wj",
               "cacheKeyParameters": [
                  "method.request.path.proxy"
               ],
               "type": "aws_proxy"
            }
         }
      }
   },
   "servers": [
      {
         "url": "https://gy415nuibc.execute-api.us-east-1.amazonaws.com/{basePath}",
         "variables": {
            "basePath": {
              "default": "/testStage"
            }
         }
      }
   ]
}
```

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2016-09-12T17:50:37Z",
    "title": "ProxyIntegrationWithLambda"
```

```
  },
  "host": "gy415nuibc.execute-api.us-east-1.amazonaws.com",
  "basePath": "/testStage",
  "schemes": [
    "https"
  ],
  "paths": {
    "/{proxy+}": {
      "x-amazon-apigateway-any-method": {
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "proxy",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {},
        "x-amazon-apigateway-integration": {
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789012:function:SimpleLambda4ProxyResource/
invocations",
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST",
          "cacheNamespace": "roq9wj",
          "cacheKeyParameters": [
            "method.request.path.proxy"
          ],
          "type": "aws_proxy"
        }
      }
    }
  }
}
```

Nell'integrazione del proxy Lambda, in fase di esecuzione, API Gateway mappa una richiesta in entrata nel `event` parametro di input della funzione Lambda. L'input include il metodo della richiesta, il percorso, le intestazioni, qualsiasi parametro della stringa di query, qualsiasi payload, il contesto associato e qualsiasi variabile di fase definita. Il formato di input viene descritto in Formato di input di una funzione Lambda per l'integrazione proxy. APIAffinché Gateway mappi correttamente l'output Lambda alle HTTP risposte, la funzione Lambda deve restituire il risultato nel formato descritto in. Formato di output di una funzione Lambda per l'integrazione proxy

Con l'integrazione proxy Lambda di una risorsa proxy tramite il metodo `ANY`, la singola funzione Lambda back-end funge da gestore eventi per tutte le richieste attraverso la risorsa proxy. Ad esempio, per registrare i modelli di traffico, è possibile fare in modo che un dispositivo mobile invii informazioni sulla posizione dello stato, della città, della via e dell'edificio inviando una richiesta `/state/city/street/house` nel URL percorso della risorsa proxy. La funzione Lambda di backend può quindi analizzare URL il percorso e inserire le tuple di posizione in una tabella DynamoDB.

Configura integrazioni personalizzate Lambda in Gateway API

Per mostrare come configurare l'integrazione personalizzata Lambda, creiamo un API gateway API per esporre il `GET /greeting?greeter={name}` metodo per richiamare una funzione Lambda. Usa uno dei seguenti esempi di funzioni Lambda. API

Usa uno dei seguenti esempi di funzioni Lambda:

Node.js

```
export const handler = function(event, context, callback) {
    var res ={
        "statusCode": 200,
        "headers": {
            "Content-Type": "*/*"
        }
    };
    if (event.greeter==null) {
        callback(new Error('Missing the required greeter parameter.'));
    } else if (event.greeter === "") {
        res.body = "Hello, World";
        callback(null, res);
    } else {
        res.body = "Hello, " + event.greeter +"!";
        callback(null, res);
    }
```

```
};
```

Python

```
import json


def lambda_handler(event, context):
    print(event)
    res = {
        "statusCode": 200,
        "headers": {
            "Content-Type": "*/*"
        }
    }

    if event['greeter'] == "":
        res['body'] = "Hello, World"
    elif (event['greeter']):
        res['body'] = "Hello, " + event['greeter'] + "!"
    else:
        raise Exception('Missing the required greeter parameter.')

    return res
```

Questa funzione risponde con un messaggio `"Hello, {name}!"` se il valore del parametro `greeter` è una stringa non vuota. La funzione restituisce un messaggio `"Hello, World!"` se il valore di `greeter` è una stringa vuota. La funzione restituisce un messaggio di errore `"Missing the required greeter parameter."` se il parametro greeter non è impostato nella richiesta in ingresso. Denominiamo la funzione `HelloWorld`.

Puoi crearla nella console Lambda o tramite la AWS CLI. In questa sezione, facciamo riferimento a questa funzione utilizzando quanto segue: ARN

```
arn:aws:lambda:us-east-1:123456789012:function:HelloWorld
```

Con la funzione Lambda impostata nel backend, procedi con la configurazione di. API

Per configurare l'integrazione personalizzata Lambda utilizzando AWS CLI

1. Chiama il `create-rest-api` comando per creare unAPI:

```
aws apigateway create-rest-api --name 'HelloWorld (AWS CLI)' --region us-west-2
```

Nota il `id` valore risultante API (`te6si5ach7`) nella risposta:

```
{
    "name": "HelloWorld (AWS CLI)",
    "id": "te6si5ach7",
    "createdDate": 1508461860
}
```

Hai bisogno di API `id` tutta questa sezione.

2. Chiama il comando `get-resources` per ottenere il valore di `id` della risorsa root:

```
aws apigateway get-resources --rest-api-id te6si5ach7 --region us-west-2
```

La risposta di esito positivo sarà simile alla seguente:

```
{
    "items": [
        {
            "path": "/",
            "id": "krznpq9xpg"
        }
    ]
}
```

Prendi nota del valore di `id` (`krznpq9xpg`) della risorsa root. Ti servirà nella prossima fase e successivamente.

3. Chiama `create-resource` per creare una risorsa API gateway di`/greeting`:

```
aws apigateway create-resource --rest-api-id te6si5ach7 \
      --region us-west-2 \
      --parent-id krznpq9xpg \
      --path-part greeting
```

La risposta con esito positivo è simile a quella riportata di seguito.

```
{
```

```
    "path": "/greeting",
    "pathPart": "greeting",
    "id": "2jf6xt",
    "parentId": "krznpq9xpg"
}
```

Prendi nota del valore `greeting` della risorsa `id` (`2jf6xt`). Ti servirà per creare un metodo nella risorsa `/greeting` nella prossima fase.

4. Chiama `put-method` per creare una richiesta di API metodo di`GET /greeting?greeter={name}`:

```
aws apigateway put-method --rest-api-id te6si5ach7 \
       --region us-west-2 \
       --resource-id 2jf6xt \
       --http-method GET \
       --authorization-type "NONE" \
       --request-parameters method.request.querystring.greeter=false
```

La risposta con esito positivo è simile a quella riportata di seguito.

```
{
    "apiKeyRequired": false,
    "httpMethod": "GET",
    "authorizationType": "NONE",
    "requestParameters": {
        "method.request.querystring.greeter": false
    }
}
```

Questo API metodo consente al client di ricevere un saluto dalla funzione Lambda nel backend. Il parametro `greeter` è facoltativo perché il back-end deve gestire un chiamante anonimo o un chiamante autoidentificato.

5. Chiama `put-method-response` per configurare la risposta `200 OK` alla richiesta del metodo `GET /greeting?greeter={name}`:

```
aws apigateway put-method-response \
        --region us-west-2 \
        --rest-api-id te6si5ach7 \
        --resource-id 2jf6xt \
        --http-method GET \
```

```
        --status-code 200
```

6. Chiama `put-integration` per configurare l'integrazione del metodo `GET /greeting?greeter={name}` con una funzione Lambda, denominata `HelloWorld`. La funzione risponde alla richiesta con un messaggio `"Hello, {name}!"` se è specificato il parametro `greeter`, altrimenti con `"Hello, World!"` se il parametro della stringa di query non è impostato.

```
aws apigateway put-integration \
        --region us-west-2 \
        --rest-api-id te6si5ach7 \
        --resource-id 2jf6xt \
        --http-method GET \
        --type AWS \
        --integration-http-method POST \
        --uri arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789012:function:HelloWorld/invocations \
        --request-templates '{"application/json":"{\"greeter\":
\"$input.params('greeter')\"}"}' \
        --credentials arn:aws:iam::123456789012:role/apigAwsProxyRole
```

Il modello di mappatura fornito qui traduce il parametro della stringa di `greeter` query nella proprietà del `greeter` payload. JSON Questo è necessario perché l'input di una funzione Lambda deve essere espresso nel corpo.

> **Important**
>
> Per le integrazioni Lambda, è necessario utilizzare il HTTP metodo di `POST` per la richiesta di integrazione, in base alla specifica dell'azione del servizio Lambda per le chiamate di funzioni. Il `uri` parametro è l'azione di richiamo della funzione. ARN Un output corretto è simile al seguente:

```
{
    "passthroughBehavior": "WHEN_NO_MATCH",
    "cacheKeyParameters": [],
    "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789012:function:HelloWorld/invocations",
    "httpMethod": "POST",
    "requestTemplates": {
```

```
        "application/json": "{\"greeter\":\"$input.params('greeter')\"}"
    },
    "cacheNamespace": "krznpq9xpg",
    "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
    "type": "AWS"
}
```

Il IAM ruolo delle politiche `apigAwsProxyRole` indispensabili che consentono al `apigateway` servizio di richiamare le funzioni Lambda. Invece di fornire un IAM ruolo per`credentials`, puoi chiamare il comando add-permission per aggiungere autorizzazioni basate sulle risorse. Ecco come la console Gateway aggiunge queste autorizzazioni. API

7. Chiama `put-integration-response` per configurare la risposta di integrazione in modo da passare l'output della funzione Lambda al client come risposta del metodo `200 OK`.

```
aws apigateway put-integration-response \
       --region us-west-2 \
       --rest-api-id te6si5ach7 \
       --resource-id 2jf6xt \
       --http-method GET \
       --status-code 200 \
       --selection-pattern ""
```

Impostando il modello di selezione su una stringa vuota, la risposta predefinita è `200 OK`.

La risposta di esito positivo sarà simile alla seguente:

```
{
    "selectionPattern": "",
    "statusCode": "200"
}
```

8. Chiama `create-deployment` per distribuirli API su una `test` fase:

```
aws apigateway create-deployment --rest-api-id te6si5ach7 --stage-name test --
region us-west-2
```

9. Prova l'APIutilizzo del seguente URL comando c in un terminale:

```
curl -X GET 'https://te6si5ach7.execute-api.us-west-2.amazonaws.com/test/greeting?
greeter=me' \
```

```
  -H 'authorization: AWS4-HMAC-SHA256 Credential={access_key}/20171020/us-
west-2/execute-api/aws4_request, SignedHeaders=content-type;host;x-amz-date,
 Signature=f327...5751'
```

Configurazione della chiamata asincrona della funzione Lambda back-end

Nell'integrazione non proxy (personalizzata) Lambda la funzione Lambda back-end viene richiamata in modo sincrono per impostazione predefinita. Questo è il comportamento desiderato per la maggior parte REST API delle operazioni. Alcune applicazioni, tuttavia, richiedono che la chiamata venga effettuata in modo asincrono, ad esempio un'operazione batch o una con latenza elevata, in genere da un componente di back-end separato. In questo caso, la funzione Lambda di backend viene richiamata in modo asincrono e il metodo REST API front-end non restituisce il risultato.

Puoi configurare la funzione Lambda affinché un'integrazione non proxy Lambda venga richiamata in modo asincrono specificando `'Event'` come tipo di chiamata Lambda. Questo avviene così:

Configurare la chiamata asincrona Lambda nella console Gateway API

Affinché tutte le invocazioni siano asincrone:

- In Richiesta di integrazione aggiungi un'intestazione `X-Amz-Invocation-Type` con un valore statico `'Event'`.

Affinché i client decidano se le invocazioni sono asincrone o sincrone:

1. In Richiesta metodo aggiungi un'intestazione `InvocationType`.
2. In Richiesta di integrazione aggiungi un'intestazione `X-Amz-Invocation-Type` con un'espressione di mappatura `method.request.header.InvocationType`.
3. I client possono includere l'`InvocationType: Event`intestazione nelle API richieste di chiamate asincrone o di chiamate sincrone. `InvocationType: RequestResponse`

Configurare la chiamata asincrona Lambda utilizzando Open API

Affinché tutte le invocazioni siano asincrone:

- Aggiungi l'intestazione alla sezione. `X-Amz-Invocation-Type` x-amazon-apigateway-integration

```
"x-amazon-apigateway-integration" : {
          "type" : "aws",
          "httpMethod" : "POST",
          "uri" : "arn:aws:apigateway:us-east-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-2:123456789012:function:my-function/invocations",
          "responses" : {
            "default" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.header.X-Amz-Invocation-Type" : "'Event'"
          },
          "passthroughBehavior" : "when_no_match",
          "contentHandling" : "CONVERT_TO_TEXT"
        }
```

Affinché i client decidano se le invocazioni sono asincrone o sincrone:

1. Aggiungi l'intestazione seguente su qualsiasi oggetto Open API Path Item.

```
"parameters" : [ {
"name" : "InvocationType",
"in" : "header",
"schema" : {
  "type" : "string"
}
} ]
```

2. Aggiungi l'`X-Amz-Invocation-Type`intestazione alla sezione. x-amazon-apigateway-integration

```
"x-amazon-apigateway-integration" : {
          "type" : "aws",
          "httpMethod" : "POST",
          "uri" : "arn:aws:apigateway:us-east-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-2:123456789012:function:my-function/invocations",
          "responses" : {
            "default" : {
              "statusCode" : "200"
            }
```

```
          },
          "requestParameters" : {
            "integration.request.header.X-Amz-Invocation-Type" :
  "method.request.header.InvocationType"
          },
          "passthroughBehavior" : "when_no_match",
          "contentHandling" : "CONVERT_TO_TEXT"
        }
```

3. I client possono includere l'`InvocationType: Event`intestazione nelle API richieste di chiamate asincrone o di chiamate sincrone. `InvocationType: RequestResponse`

Configurare la chiamata asincrona Lambda utilizzando AWS CloudFormation

I seguenti AWS CloudFormation modelli mostrano come configurare le chiamate asincrone. `AWS::ApiGateway::Method`

Affinché tutte le invocazioni siano asincrone:

```
AsyncMethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref AsyncResource
      HttpMethod: GET
      ApiKeyRequired: false
      AuthorizationType: NONE
      Integration:
        Type: AWS
        RequestParameters:
          integration.request.header.X-Amz-Invocation-Type: "'Event'"
        IntegrationResponses:
            - StatusCode: '200'
        IntegrationHttpMethod: POST
        Uri: !Sub arn:aws:apigateway:${AWS::Region}:lambda:path/2015-03-31/functions/
${myfunction.Arn}$/invocations
      MethodResponses:
        - StatusCode: '200'
```

Affinché i client decidano se le invocazioni sono asincrone o sincrone:

```
AsyncMethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref AsyncResource
      HttpMethod: GET
      ApiKeyRequired: false
      AuthorizationType: NONE
      RequestParameters:
        method.request.header.InvocationType: false
      Integration:
        Type: AWS
        RequestParameters:
          integration.request.header.X-Amz-Invocation-Type:
 method.request.header.InvocationType
        IntegrationResponses:
            - StatusCode: '200'
        IntegrationHttpMethod: POST
        Uri: !Sub arn:aws:apigateway:${AWS::Region}:lambda:path/2015-03-31/functions/
${myfunction.Arn}$/invocations
      MethodResponses:
        - StatusCode: '200'
```

I client possono includere l'`InvocationType: Event`intestazione nelle API richieste di chiamate asincrone o di chiamate sincrone. `InvocationType: RequestResponse`

Gestione degli errori Lambda in API Gateway

Per le integrazioni personalizzate Lambda, devi mappare gli errori restituiti da Lambda nella risposta di integrazione alle risposte di errore HTTP standard per i client. Altrimenti, gli errori Lambda vengono restituiti come risposte `200 OK` per impostazione predefinita e il risultato non è intuitivo per gli utenti dell'API.

Esistono due tipi di errore che Lambda può restituire: errori standard ed errori personalizzati. Nella tua API, devi gestirli in modo diverso.

Con l'integrazione proxy Lambda, Lambda deve restituire un output nel formato seguente:

```
{
  "isBase64Encoded" : "boolean",
  "statusCode": "number",
```

```
  "headers": { ... },
  "body": "JSON string"
}
```

In questo output, `statusCode` è in genere `4XX` per un errore del client e `5XX` per un errore del server. API Gateway gestisce questi errori mappando l'errore Lambda a una risposta di errore HTTP, in base al `statusCode` specificato. Affinché API Gateway passi il tipo di errore (ad esempio, `InvalidParameterException`), come parte della risposta al client, la funzione Lambda deve includere un'intestazione (ad esempio, `"X-Amzn-ErrorType":"InvalidParameterException"`) nella proprietà `headers`.

Argomenti

- Gestione degli errori Lambda standard in API Gateway
- Gestione degli errori Lambda personalizzati in API Gateway

Gestione degli errori Lambda standard in API Gateway

Un errore AWS Lambda standard ha il seguente formato:

```
{
  "errorMessage": "<replaceable>string</replaceable>",
  "errorType": "<replaceable>string</replaceable>",
  "stackTrace": [
    "<replaceable>string</replaceable>",
    ...
  ]
}
```

Qui, `errorMessage` è un'espressione della stringa dell'errore. `errorType` è un tipo di eccezione o di errore dipendente dal linguaggio. `stackTrace` è un elenco di espressioni delle stringhe che mostrano la traccia dello stack che porta al verificarsi dell'errore.

Si consideri, ad esempio, la seguente funzione Lambda JavaScript (Node.js).

```
export const handler = function(event, context, callback) {
    callback(new Error("Malformed input ..."));
};
```

Questa funzione restituisce il seguente errore Lambda standard, contenente `Malformed input ...` come messaggio di errore:

```
{
  "errorMessage": "Malformed input ...",
  "errorType": "Error",
  "stackTrace": [
    "export const handler (/var/task/index.js:3:14)"
  ]
}
```

Allo stesso modo, considera la seguente funzione Python Lambda che genera un'eccezione `Exception` con lo stesso messaggio di errore `Malformed input ...`.

```
def lambda_handler(event, context):
    raise Exception('Malformed input ...')
```

La funzione restituisce il seguente errore Lambda standard:

```
{
  "stackTrace": [
    [
      "/var/task/lambda_function.py",
      3,
      "lambda_handler",
      "raise Exception('Malformed input ...')"
    ]
  ],
  "errorType": "Exception",
  "errorMessage": "Malformed input ..."
}
```

Da notare che i valori delle proprietà `errorType` e `stackTrace` dipendono dal linguaggio. L'errore standard si applica anche a qualsiasi oggetto di errore che è un'estensione dell'oggetto `Error` o una sottoclasse della classe `Exception`.

Per mappare l'errore Lambda standard a una risposta del metodo, devi prima stabilire il codice di stato HTTP per un dato errore Lambda. Devi quindi impostare un modello di espressione regolare per la proprietà `selectionPattern` di IntegrationResponse associata al codice di stato HTTP specificato. Nella console Gateway API, questo `selectionPattern` è indicato come Espressione regolare errore Lambda sotto ogni risposta di integrazione nella sezione Risposta di integrazione.

> **Note**
>
> API Gateway usa le espressioni regolari basate sullo stile del modello Java per la mappatura della risposta. Per ulteriori informazioni, consulta Pattern nella documentazione Oracle.

Ad esempio, per configurare una nuova espressione `selectionPattern` tramite AWS CLI, è necessario chiamare il seguente comando put-integration-response:

```
aws apigateway put-integration-response --rest-api-id z0vprf0mdh --resource-id x3o5ih
 --http-method GET --status-code 400 --selection-pattern "Malformed.*" --region us-
west-2
```

Accertati di configurare anche il codice di errore corrispondente (`400`) sulla risposta del metodo. Altrimenti, API Gateway restituisce una risposta di errore di configurazione non valida al runtime.

> **Note**
>
> Al runtime, API Gateway abbina l'elemento `errorMessage` dell'errore Lambda al modello dell'espressione regolare per la proprietà `selectionPattern`. Se c'è una corrispondenza, API Gateway restituisce l'errore Lambda come risposta HTTP del codice di stato HTTP corrispondente. Se non c'è corrispondenza, API Gateway restituisce l'errore come risposta predefinita oppure genera un'eccezione di configurazione non valida se non è stata configurata una risposta predefinita.
> Impostare il valore `selectionPattern` su `.*` per una data risposta significa reimpostare questa risposta come risposta predefinita. Questo succede perché tale modello di selezione corrisponderà a tutti i messaggi di errore, incluso null, come, ad esempio, messaggi di errore non specificati. La mappatura che ne deriva sovrascrive la mappatura predefinita.

Per aggiornare un valore `selectionPattern` esistente tramite AWS CLI, chiama l'operazione integrationresponse:update per sostituire il valore del percorso `/selectionPattern` con l'espressione regex specifica del modello `Malformed*`.

Per impostare l'espressione `selectionPattern` tramite la console Gateway API, immetti l'espressione nella casella di testo Espressione regolare errore Lambda quando configuri o aggiorni una risposta di integrazione di un codice di stato HTTP specifico.

Gestione degli errori Lambda personalizzati in API Gateway

Al posto dell'errore standard descritto nella sezione precedente, AWS Lambda consente di restituire un oggetto errore personalizzato come stringa JSON. L'errore può essere qualsiasi oggetto JSON valido. Ad esempio, la seguente funzione Lambda JavaScript (Node.js) restituisce un errore personalizzato:

```
export const handler = (event, context, callback) => {
    ...
    // Error caught here:
    var myErrorObj = {
        errorType : "InternalServerError",
        httpStatus : 500,
        requestId : context.awsRequestId,
        trace : {
            "function": "abc()",
            "line": 123,
            "file": "abc.js"
        }
    }
    callback(JSON.stringify(myErrorObj));
};
```

Devi trasformare l'oggetto `myErrorObj` in una stringa JSON prima di chiamare `callback` per uscire dalla funzione. In caso contrario, viene restituito `myErrorObj` come stringa di `"[object Object]"`. Quando un metodo dell'API è integrato con la precedente funzione Lambda, API Gateway riceve una risposta di integrazione con il seguente payload:

```
{
    "errorMessage": "{\"errorType\":\"InternalServerError\",\"httpStatus\":500,
\"requestId\":\"e5849002-39a0-11e7-a419-5bb5807c9fb2\",\"trace\":{\"function\":
\"abc()\",\"line\":123,\"file\":\"abc.js\"}}"
}
```

Come nel caso delle risposte di integrazione, puoi passare questa risposta di errore invariata alla risposta di metodo. Oppure puoi avere un modello di mappatura per trasformare il payload in un formato diverso. Ad esempio, considera il seguente modello di mappatura del corpo per la risposta di un metodo del codice di stato `500`:

```
{
```

```
    errorMessage: $input.path('$.errorMessage');
}
```

Questo modello traduce il corpo della risposta di integrazione che contiene la stringa JSON di errore personalizzato nel corpo di risposta del metodo seguente. Questo corpo di risposta del metodo contiene un oggetto JSON di errore personalizzato:

```
{
    "errorMessage" : {
        errorType : "InternalServerError",
        httpStatus : 500,
        requestId : context.awsRequestId,
        trace : {
            "function": "abc()",
            "line": 123,
            "file": "abc.js"
        }
    }
};
```

In base ai tuoi requisiti API, potresti aver bisogno di passare alcune o tutte le proprietà degli errori personalizzati come parametri dell'intestazione della risposta di metodo. Puoi farlo applicando le mappature degli errori personalizzati dal corpo della risposta di integrazione alle intestazioni della risposta di metodo.

Ad esempio, la seguente estensione OpenAPI definisce una mappatura dalle proprietà `errorMessage.errorType`, `errorMessage.httpStatus`, `errorMessage.trace.function` ed `errorMessage.trace` alle intestazioni `error_type`, `error_status`, `error_trace_function` ed `error_trace`, rispettivamente.

```
"x-amazon-apigateway-integration": {
    "responses": {
        "default": {
          "statusCode": "200",
          "responseParameters": {
            "method.response.header.error_trace_function":
 "integration.response.body.errorMessage.trace.function",
            "method.response.header.error_status":
 "integration.response.body.errorMessage.httpStatus",
            "method.response.header.error_type":
 "integration.response.body.errorMessage.errorType",
```

```
            "method.response.header.error_trace":
 "integration.response.body.errorMessage.trace"
          },
          ...
        }
    }
}
```

Al runtime, API Gateway deserializza il parametro `integration.response.body` nel corso delle mappature delle intestazioni. Tuttavia, questa deserializzazione si applica solo alle mappature corpo-intestazione per le risposte di errore API Gateway personalizzate e non si applica alle mappature corpo-corpo tramite `$input.body`. Con queste mappature corpo-errore-personalizzato-a-intestazione, il client riceve le seguenti intestazioni come parte della risposta di metodo, a condizione che le intestazioni `error_status`, `error_trace`, `error_trace_function` e `error_type` siano dichiarate nella richiesta di metodo.

```
"error_status":"500",
"error_trace":"{\"function\":\"abc()\",\"line\":123,\"file\":\"abc.js\"}",
"error_trace_function":"abc()",
"error_type":"InternalServerError"
```

La proprietà `errorMessage.trace` del corpo della risposta di integrazione è una proprietà complessa. Viene mappata all'intestazione `error_trace` come una stringa JSON.

Integrazioni HTTP per le API REST in API Gateway

Puoi integrare un metodo API con un endpoint HTTP tramite l'integrazione proxy di HTTP o l'integrazione personalizzata di HTTP.

API Gateway supporta le seguenti porte dell'endpoint: 80, 443 e 1024-65535.

Con l'integrazione proxy la configurazione è semplice. Devi solo impostare il metodo HTTP e l'URI dell'endpoint HTTP, in base ai requisiti del back-end, se non sei interessato alla codifica o al caching dei contenuti.

Con l'integrazione personalizzata la configurazione è più complessa. Oltre alla procedura di configurazione dell'integrazione proxy, devi specificare come verranno mappati i dati della richiesta in ingresso alla richiesta di integrazione e i dati della risposta di integrazione risultante alla risposta del metodo.

Argomenti

- Configurazione di integrazioni proxy HTTP in API Gateway
- Configurazione di integrazioni personalizzate HTTP in API Gateway

Configurazione di integrazioni proxy HTTP in API Gateway

Per configurare una risorsa proxy con il tipo di integrazione proxy HTTP, crea una risorsa API con un parametro di percorso greedy, ad esempio `/parent/{proxy+}`, e integra la risorsa con un endpoint di back-end HTTP, ad esempio `https://petstore-demo-endpoint.execute-api.com/petstore/{proxy}`, nel metodo `ANY`. Il parametro di percorso greedy deve trovarsi alla fine del percorso della risorsa.

Come per una risorsa non proxy, puoi configurare una risorsa proxy con l'integrazione proxy HTTP usando la console API Gateway, importando un file di definizione OpenAPI o chiamando direttamente l'API REST di API Gateway. Per istruzioni dettagliate sull'uso della console API Gateway per configurare una risorsa proxy con l'integrazione HTTP, consulta Tutorial: creare un'API REST con un'integrazione proxy HTTP.

Il seguente file di definizione OpenAPI mostra un esempio di API con una risorsa proxy integrata con il PetStoresito Web.

OpenAPI 3.0

```
{
   "openapi": "3.0.0",
   "info": {
      "version": "2016-09-12T23:19:28Z",
      "title": "PetStoreWithProxyResource"
   },
   "paths": {
      "/{proxy+}": {
         "x-amazon-apigateway-any-method": {
            "parameters": [
               {
                  "name": "proxy",
                  "in": "path",
                  "required": true,
                  "schema": {
                     "type": "string"
                  }
               }
            ],
```

```
            "responses": {},
            "x-amazon-apigateway-integration": {
               "responses": {
                  "default": {
                     "statusCode": "200"
                  }
               },
               "requestParameters": {
                  "integration.request.path.proxy": "method.request.path.proxy"
               },
               "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/
{proxy}",
               "passthroughBehavior": "when_no_match",
               "httpMethod": "ANY",
               "cacheNamespace": "rbftud",
               "cacheKeyParameters": [
                  "method.request.path.proxy"
               ],
               "type": "http_proxy"
            }
         }
      }
   },
   "servers": [
      {
         "url": "https://4z9giyi2c1.execute-api.us-east-1.amazonaws.com/{basePath}",
         "variables": {
            "basePath": {
              "default": "/test"
            }
         }
      }
   ]
}
```

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2016-09-12T23:19:28Z",
    "title": "PetStoreWithProxyResource"
  },
```

```
  "host": "4z9giyi2c1.execute-api.us-east-1.amazonaws.com",
  "basePath": "/test",
  "schemes": [
    "https"
  ],
  "paths": {
    "/{proxy+}": {
      "x-amazon-apigateway-any-method": {
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "proxy",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {},
        "x-amazon-apigateway-integration": {
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.path.proxy": "method.request.path.proxy"
          },
          "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/{proxy}",
          "passthroughBehavior": "when_no_match",
          "httpMethod": "ANY",
          "cacheNamespace": "rbftud",
          "cacheKeyParameters": [
            "method.request.path.proxy"
          ],
          "type": "http_proxy"
        }
      }
    }
  }
}
```

In questo esempio, viene dichiarata una chiave cache nel parametro del percorso `method.request.path.proxy` della risorsa proxy. Questa è l'impostazione predefinita quando crei l'API tramite la console API Gateway. Il percorso di base dell'API (`/test`corrispondente a una fase) è mappato alla PetStore pagina del sito Web (`/petstore`). La singola richiesta di integrazione rispecchia l'intero PetStore sito Web utilizzando la variabile greedy path dell'API e il metodo `ANY` catch-all. Negli esempi seguenti viene illustrato questo mirroring.

- Imposta **`ANY`** come **`GET`** e **`{proxy+}`** come **`pets`**

  Richiesta di metodo iniziata dal front-end:

  ```
  GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets HTTP/1.1
  ```

  Richiesta di integrazione inviata al back-end:

  ```
  GET http://petstore-demo-endpoint.execute-api.com/petstore/pets HTTP/1.1
  ```

  Le istanze run-time del metodo `ANY` e le risorse proxy sono entrambe valide. La chiamata restituisce una risposta `200 OK` con il payload contenente il primo batch di animali, nel modo in cui viene restituito dal back-end.

- Imposta **`ANY`** come **`GET`** e **`{proxy+}`** come **`pets?type=dog`**

  ```
  GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets?type=dog
   HTTP/1.1
  ```

  Richiesta di integrazione inviata al back-end:

  ```
  GET http://petstore-demo-endpoint.execute-api.com/petstore/pets?type=dog HTTP/1.1
  ```

  Le istanze run-time del metodo `ANY` e le risorse proxy sono entrambe valide. La chiamata restituisce una risposta `200 OK` con il payload contenente il primo batch di cani specifici, nel modo in cui viene restituito dal back-end.

- Imposta **`ANY`** come **`GET`** e **`{proxy+}`** come **`pets/{petId}`**

  Richiesta di metodo iniziata dal front-end:

  ```
  GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets/1 HTTP/1.1
  ```

Richiesta di integrazione inviata al back-end:

```
GET http://petstore-demo-endpoint.execute-api.com/petstore/pets/1 HTTP/1.1
```

Le istanze run-time del metodo `ANY` e le risorse proxy sono entrambe valide. La chiamata restituisce una risposta `200 OK` con il payload contenente l'animale specificato, nel modo in cui viene restituito dal back-end.

- Imposta **`ANY`** come **`POST`** e **`{proxy+}`** come **`pets`**

  Richiesta di metodo iniziata dal front-end:

```
POST https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets HTTP/1.1
Content-Type: application/json
Content-Length: ...

{
  "type" : "dog",
  "price" : 1001.00
}
```

  Richiesta di integrazione inviata al back-end:

```
POST http://petstore-demo-endpoint.execute-api.com/petstore/pets HTTP/1.1
Content-Type: application/json
Content-Length: ...

{
  "type" : "dog",
  "price" : 1001.00
}
```

  Le istanze run-time del metodo `ANY` e le risorse proxy sono entrambe valide. La chiamata restituisce una risposta `200 OK` con il payload contenente il nuovo animale creato, nel modo in cui viene restituito dal back-end.

- Imposta **`ANY`** come **`GET`** e **`{proxy+}`** come **`pets/cat`**

  Richiesta di metodo iniziata dal front-end:

```
GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets/cat
```

Richiesta di integrazione inviata al back-end:

```
GET http://petstore-demo-endpoint.execute-api.com/petstore/pets/cat
```

L'istanza run-time del percorso della risorsa del proxy non corrisponde all'endpoint di un back-end e la relativa richiesta non è valida. Di conseguenza, viene restituita una risposta `400 Bad Request` con il seguente messaggio di errore.

```
{
  "errors": [
    {
      "key": "Pet2.type",
      "message": "Missing required field"
    },
    {
      "key": "Pet2.price",
      "message": "Missing required field"
    }
  ]
}
```

- Imposta **`ANY`** come **`GET`** e **`{proxy+}`** come **`null`**

  Richiesta di metodo iniziata dal front-end:

  ```
  GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test
  ```

  Richiesta di integrazione inviata al back-end:

  ```
  GET http://petstore-demo-endpoint.execute-api.com/petstore/pets
  ```

  La risorsa di destinazione è il padre della risorsa del proxy, ma l'istanza run-time del metodo `ANY` non è definita nell'API di quella risorsa. Di conseguenza, la richiesta `GET` restituisce una risposta `403 Forbidden` con il messaggio di errore `Missing Authentication Token` restituito da API Gateway. Se l'API espone il metodo `ANY` o `GET` alla risorsa padre (`/`), la chiamata restituisce una risposta `404 Not Found` con il messaggio `Cannot GET /petstore` restituito dal back-end.

Per qualsiasi richiesta del client, se l'URL dell'endpoint di destinazione non è valido o il verbo HTTP è valido ma non è supportato, il back-end restituisce una risposta `404 Not Found`. Per un metodo HTTP non supportato, viene restituita una risposta `403 Forbidden`.

Configurazione di integrazioni personalizzate HTTP in API Gateway

Con l'integrazione HTTP personalizzata, hai più controllo su quali dati passare tra un metodo API e un'integrazione API e su come passarli. Puoi fare questo utilizzando le mappature dei dati.

Come parte della configurazione della richiesta del metodo, imposti la proprietà requestParameters su una risorsa Method. Questo dichiara quali parametri della richiesta di metodo, forniti dal client, devono essere mappati ai parametri di richiesta di integrazione o alle proprietà del corpo applicabili prima di essere inviati al back-end. Quindi, come parte della configurazione della richiesta di integrazione, impostate la proprietà RequestParameters sulla risorsa di integrazione corrispondente per specificare le parameter-to-parameter mappature. Imposti inoltre la proprietà requestTemplates per specificare i modelli di mappatura, uno per ogni tipo di contenuto supportato. I modelli di mappatura mappano i parametri della richiesta di metodo, o il corpo, al corpo della richiesta di integrazione.

Analogamente, come parte della configurazione della risposta del metodo, impostate la proprietà ResponseParameters sulla risorsa. MethodResponse Questo dichiara quali parametri della risposta di metodo, da inviare al client, devono essere mappati dai parametri della risposta di integrazione o da alcune proprietà del corpo applicabili restituite dal back-end. Quindi, come parte della configurazione della risposta di integrazione, impostate la proprietà ResponseParameters sulla risorsa IntegrationResponsecorrispondente per specificare le mappature. parameter-to-parameter Imposti inoltre la mappa responseTemplates per specificare i modelli di mappatura, uno per ogni tipo di contenuto supportato. I modelli di mappatura mappano i parametri della risposta di integrazione, o le proprietà del corpo della risposta di integrazione, al corpo della risposta del metodo.

Per ulteriori informazioni sulla configurazione dei modelli di mappatura, consulta Trasformazioni dei dati per REST APIs in Gateway API.

Integrazioni private per REST APIs in API Gateway

L'integrazione privata di API Gateway semplifica l'esposizione delle tueHTTP/HTTPSrisorse all'interno di un Amazon VPC per l'accesso da parte di clienti esterni a. VPC Per estendere l'accesso alle tue VPC risorse private oltre i VPC confini, puoi creare un'integrazione API con private. Puoi controllare l'accesso al tuo dispositivo API utilizzando uno qualsiasi dei metodi di autorizzazione supportati da API Gateway.

Per creare un'integrazione privata, è necessario innanzitutto creare un collegamento Network Load Balancer. Il Network Load Balancer deve disporre di un listener che indirizza le richieste alle risorse del tuo. VPC Per migliorare la disponibilità del tuoAPI, assicurati che il Network Load Balancer indirizzi il traffico verso le risorse in più di una zona di disponibilità nel. Regione AWS Quindi, crei un VPC link da utilizzare per connettere il tuo Network Load Balancer API e il tuo Network Load Balancer. Dopo aver creato un VPC link, crei integrazioni private per indirizzare il traffico dalle tue risorse API alle risorse presenti nel tuo sito VPC tramite il tuo VPC link e Network Load Balancer.

> **Note**
>
> Il Network Load Balancer API deve appartenere allo stesso AWS account.

Con l'integrazione API Gateway private, puoi abilitare l'accesso aHTTP/HTTPSrisorse all'interno di un ambiente VPC senza una conoscenza dettagliata delle configurazioni di rete private o dei dispositivi tecnologici specifici.

Argomenti

- Configurare un Network Load Balancer per le integrazioni private del API gateway
- Concedi a API Gateway le autorizzazioni per creare un collegamento VPC
- Configura un API Gateway API con integrazioni private utilizzando il AWS CLI
- Configura API con integrazioni private utilizzando Open API
- APIAccount gateway utilizzati per le integrazioni private

Configurare un Network Load Balancer per le integrazioni private del API gateway

La procedura seguente descrive i passaggi per configurare un Network Load Balancer NLB () API per le integrazioni private di Gateway utilizzando la console EC2 Amazon e fornisce riferimenti per istruzioni dettagliate per ogni passaggio.

Per ogni risorsa in VPC cui sono presenti, è sufficiente configurarne una NLB alla volta. VPCLink NLBSupporta più ascoltatori e gruppi target per. NLB È possibile configurare ogni servizio come listener specifico su NLB e utilizzarne uno VPCLink per connettersi a. NLB Quando si crea l'integrazione privata in API Gateway, si definisce quindi ogni servizio utilizzando la porta specifica assegnata a ciascun servizio. Per ulteriori informazioni, consulta the section called “Tutorial: crea un'API REST con un'integrazione privata”.

> **Note**
>
> Il Network Load Balancer API deve appartenere allo stesso AWS account.

Per creare un Network Load Balancer per l'integrazione privata utilizzando la API console Gateway

1. Accedi a AWS Management Console e apri la EC2 console Amazon all'indirizzo https://console.aws.amazon.com/ec2/.
2. Configura un server Web su un'EC2istanza Amazon. Per un esempio di configurazione, consulta Installazione di un server LAMP Web su Amazon Linux 2.
3. Crea un Network Load Balancer, registra l'EC2istanza presso un gruppo target e aggiungi il gruppo target a un listener del Network Load Balancer. Per ulteriori informazioni, segui le istruzioni contenute in Nozioni di base sui sistemi Network Load Balancer.
4. Dopo la creazione del Network Load Balancer, procedi come segue:
   a. Nota il ARN Network Load Balancer. Ti servirà per creare un VPC collegamento in API Gateway per l'integrazione API con le VPC risorse alla base del Network Load Balancer.
   b. Disattiva la valutazione dei gruppi di sicurezza per. PrivateLink
      - Per disattivare la valutazione dei gruppi di sicurezza per il PrivateLink traffico che utilizza la console, puoi scegliere la scheda Sicurezza, quindi Modifica. Nelle impostazioni di sicurezza, deseleziona Applica le regole in entrata al PrivateLink traffico.
      - Per disattivare la valutazione del gruppo di sicurezza per il PrivateLink traffico utilizzando il AWS CLI, usa il seguente comando:

```
aws elbv2 set-security-groups --load-balancer-
arn arn:aws:elasticloadbalancing:us-east-2:111122223333:loadbalancer/net/
my-loadbalancer/abc12345 \
  --security-groups sg-123345a --enforce-security-group-inbound-rules-on-
private-link-traffic off
```

> **Note**
>
> Non aggiungete dipendenze a API Gateway CIDRs poiché sono destinate a cambiare senza preavviso.

Concedi a API Gateway le autorizzazioni per creare un collegamento VPC

Affinché tu o un utente del tuo account possiate creare e gestire un VPC link, voi o l'utente dovete disporre delle autorizzazioni per creare, eliminare e visualizzare le configurazioni dei servizi VPC endpoint, modificare le autorizzazioni dei servizi VPC endpoint ed esaminare i sistemi di bilanciamento del carico. Per concedere tali autorizzazioni, esegui la procedura illustrata di seguito.

Per concedere le autorizzazioni per creare, aggiornare ed eliminare un link VPC

1. Crea una IAM politica simile alla seguente:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "apigateway:POST",
                "apigateway:GET",
                "apigateway:PATCH",
                "apigateway:DELETE"
            ],
            "Resource": [
                "arn:aws:apigateway:us-east-1::/vpclinks",
                "arn:aws:apigateway:us-east-1::/vpclinks/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:DescribeLoadBalancers"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVpcEndpointServiceConfiguration",
                "ec2:DeleteVpcEndpointServiceConfigurations",
                "ec2:DescribeVpcEndpointServiceConfigurations",
                "ec2:ModifyVpcEndpointServicePermissions"
            ],
            "Resource": "*"
```

```
        }
    ]
}
```

2. Crea o scegli un IAM ruolo e allega al ruolo la politica precedente.
3. Assegna il IAM ruolo a te o a un utente del tuo account che sta creando VPC link.

Configura un API Gateway API con integrazioni private utilizzando il AWS CLI

Il seguente tutorial mostra come utilizzare AWS CLI per creare un VPC collegamento e un'integrazione privata. Sono richiesti i seguenti prerequisiti:

- È necessario un Network Load Balancer creato e configurato con l'VPCorigine come destinazione. Per ulteriori informazioni, consulta Configurare un Network Load Balancer per le integrazioni private del API gateway. Deve essere Account AWS uguale al tuoAPI. È necessario il Network Load Balancer ARN per creare il VPC collegamento.
- Per creare e gestire un`VpcLink`, hai bisogno delle autorizzazioni per creare un file `VpcLink` nel tuo. API Non sono necessarie le autorizzazioni per utilizzare il. `VpcLink` Per ulteriori informazioni, consulta Concedi a API Gateway le autorizzazioni per creare un collegamento VPC.

Per configurare un account API con l'integrazione privata utilizzando AWS CLI

1. Il create-vpc-linkcomando seguente crea un `VpcLink` target per il Network Load Balancer specificato.

   ```
   aws apigateway create-vpc-link \
       --name my-test-vpc-link \
       --target-arns arn:aws:elasticloadbalancing:us-east-2:123456789012:loadbalancer/
   net/my-vpclink-test-nlb/1234567890abcdef
   ```

   L'output di questo comando riconosce la ricezione della richiesta e mostra lo stato `PENDING` per il `VpcLink` in fase di creazione.

   ```
   {
       "status": "PENDING",
       "targetArns": [
           "arn:aws:elasticloadbalancing:us-east-2:123456789012:loadbalancer/net/my-
   vpclink-test-nlb/1234567890abcdef"
       ],
   ```

```
    "id": "gim7c3",
    "name": "my-test-vpc-link"
}
```

APIGateway impiega 2-4 minuti per completare la creazione di. `VpcLink` Dopo il completamento dell'operazione, il valore di `status` è `AVAILABLE`. È possibile verificarlo chiamando il seguente get-vpc-linkcomando:

```
aws apigateway get-vpc-link --vpc-link-id gim7c3
```

Se l'operazione non riesce, lo stato è `FAILED` e `statusMessage` contiene il messaggio di errore. Ad esempio, se si tenta di creare un `VpcLink` Network Load Balancer già associato a un VPC endpoint, si ottiene quanto segue sulla proprietà: `statusMessage`

```
"NLB is already associated with another VPC Endpoint Service"
```

Una volta creata correttamente, `VpcLink` è possibile crearne una API e integrarla con la VPC risorsa tramite. `VpcLink`

Nota il `id` valore del nuovo file creato`VpcLink`. In questo output di esempio, è`gim7c3`. Questo valore è necessario per configurare l'integrazione privata.

2. Il create-rest-apicomando seguente crea una `RestApi`risorsa API Gateway:

```
aws apigateway create-rest-api --name 'My VPC Link Test'
```

Annotate `RestApi` il `id` valore e il `rootResourceId` valore `RestApi` del risultato restituito. È necessario questo valore per eseguire ulteriori operazioni suAPI.

A scopo illustrativo, creeremo un metodo API con solo un `GET` metodo sulla risorsa radice (`/`) e integreremo il metodo con. `VpcLink`

3. Configura il metodo `GET /`. Utilizzate il seguente comando put-method e immettete `Id` as the `rest-api-id` e as the`rootResourceId`: `resource-id`

```
aws apigateway put-method \
       --rest-api-id  abcdef123 \
       --resource-id skpp60rab7 \
       --http-method GET \
```

```
    --authorization-type "NONE"
```

Se non si utilizza l'integrazione del proxy con`VpcLink`, è inoltre necessario impostare almeno un metodo di risposta del codice di `200` stato. In questo caso usiamo l'integrazione proxy.

4. Dopo aver creato il `GET /` metodo, configurate l'integrazione. Per un'integrazione privata, si utilizza il `connection-id` parametro per fornire l'`VpcLink`ID. È possibile utilizzare una variabile di fase o immettere direttamente l'`VpcLink`ID. Il parametro `uri` non viene usato per il routing delle richieste all'endpoint, ma viene usato per impostare l'intestazione `Host` per la convalida del certificato.

   Use the VPC link ID

   Il seguente comando put-integration utilizza l'`VpcLink`ID direttamente nell'integrazione:

```
aws apigateway put-integration \
    --rest-api-id abcdef123 \
    --resource-id skpp60rab7 \
    --uri 'http://my-vpclink-test-nlb-1234567890abcdef.us-east-2.amazonaws.com'
 \
    --http-method GET \
    --type HTTP_PROXY \
    --integration-http-method GET \
    --connection-type VPC_LINK \
    --connection-id gim7c3
```

   Use a stage variable

   Il seguente comando put-integration utilizza una variabile stage per fare riferimento all'ID del collegamento. VPC Quando lo distribuisci API su uno stage, imposti l'ID del VPC link.

```
aws apigateway put-integration \
    --rest-api-id abcdef123 \
    --resource-id skpp60rab7 \
    --uri 'http://my-vpclink-test-nlb-1234567890abcdef.us-east-2.amazonaws.com'
 \
    --http-method GET \
    --type HTTP_PROXY \
    --integration-http-method GET \
    --connection-type VPC_LINK \
    --connection-id "\${stageVariables.vpcLinkId}"
```

Assicurati di racchiudere tra virgolette doppie l'espressione della variabile di fase (`${stageVariables.vpcLinkId}`) e di aggiungere il carattere di escape davanti a `$`.

In qualsiasi momento, puoi anche aggiornare l'integrazione per modificare il`connection-id`. Il seguente comando update-integration mostra come aggiornare l'integrazione:

```
 aws apigateway update-integration \
    --rest-api-id abcdef123 \
    --resource-id skpp60rab7 \
    --http-method GET \
    --patch-operations '[{"op":"replace","path":"/
connectionId","value":"${stageVariables.vpcLinkId}"}]'
```

Assicurati di utilizzare un JSON elenco di stringhe come valore del parametro. `patch-operations`

Poiché abbiamo utilizzato l'integrazione con proxy privato, ora il tuo API è pronto per l'implementazione e per le esecuzioni di test. Con l'integrazione senza proxy, devi anche configurare il metodo di risposta e la risposta di integrazione, proprio come faresti quando configuri e APIcon integrazioni HTTP personalizzate.

5. Se hai usato la variabile stage per definire la tua`connection-id`, devi implementarla per testarlaAPI. Il seguente comando create-deployment mostra come distribuire il tuo API con una variabile stage:

```
aws apigateway create-deployment \
    --rest-api-id abcdef123 \
    --stage-name test \
    --variables vpcLinkId=gim7c3
```

Per aggiornare la variabile stage con un `VpcLink` ID diverso, ad esempio*`asf9d7`*, chiama il comando update-stage:

```
aws apigateway update-stage \
    --rest-api-id abcdef123 \
    --stage-name test \
    --patch-operations op=replace,path='/variables/vpcLinkId',value='asf9d7'
```

Quando codificate la `connection-id` proprietà con l'`VpcLink`ID letterale, non è necessario implementare il vostro per testarla. API Usa il test-invoke-methodcomando per testare il tuo API prima che venga distribuito.

6. Usa il seguente comando per richiamare il tuo: API

```
curl -X GET https://abcdef123.execute-api.us-east-2.amazonaws.com/test
```

In alternativa, puoi inserire il tuo API invoke- URL in un browser web per visualizzare il risultato.

Configura API con integrazioni private utilizzando Open API

Puoi configurarne una API con l'integrazione privata importando API il API file Open. Le impostazioni sono simili alle API definizioni Open di an API con HTTP integrazioni, con le seguenti eccezioni:

- Devi impostare esplicitamente `connectionType` su `VPC_LINK`.
- Devi impostare esplicitamente `connectionId` sull'ID di un oggetto `VpcLink` o su una variabile di fase che fa riferimento all'ID di un oggetto `VpcLink`.
- Il `uri` parametro nell'integrazione privata punta a un HTTPS endpointHTTP/inVPC, ma viene invece utilizzato per configurare l'intestazione della richiesta di integrazione. `Host`
- Il `uri` parametro nell'integrazione privata con un HTTPS endpoint in VPC viene utilizzato per verificare il nome di dominio indicato rispetto a quello del certificato installato sull'endpoint. VPC

Per fare riferimento all'ID di `VpcLink`, è possibile usare una variabile di fase. In alternativa, è possibile assegnare il valore ID direttamente a `connectionId`.

Il API file Open in JSON formato seguente mostra un esempio di un API con un VPC collegamento a cui fa riferimento una variabile di fase (): `${stageVariables.vpcLinkId}`

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2017-11-17T04:40:23Z",
    "title": "MyApiWithVpcLink"
  },
  "host": "p3wocvip9a.execute-api.us-west-2.amazonaws.com",
```

```
  "basePath": "/test",
  "schemes": [
    "https"
  ],
  "paths": {
    "/": {
      "get": {
        "produces": [
          "application/json"
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "uri": "http://my-vpclink-test-nlb-1234567890abcdef.us-
east-2.amazonaws.com",
          "passthroughBehavior": "when_no_match",
          "connectionType": "VPC_LINK",
          "connectionId": "${stageVariables.vpcLinkId}",
          "httpMethod": "GET",
          "type": "http_proxy"
        }
      }
    }
  },
  "definitions": {
    "Empty": {
      "type": "object",
      "title": "Empty Schema"
    }
  }
}
```

APIAccount gateway utilizzati per le integrazioni private

I seguenti account API Gateway specifici per regione IDs vengono aggiunti automaticamente al servizio VPC endpoint non appena `AllowedPrincipals` si crea un. `VpcLink`

Region	ID account
us-east-1	392220576650
us-east-2	718770453195
us-west-1	968246515281
us-west-2	109351309407
ca-central-1	796887884028
eu-west-1	631144002099
eu-west-2	544388816663
eu-west-3	061510835048
eu-central-1	474240146802
eu-central-2	166639821150
eu-north-1	394634713161
eu-south-1	753362059629
eu-south-2	359345898052
ap-northeast-1	969236854626
ap-northeast-2	020402002396
ap-northeast-3	360671645888
ap-southeast-1	195145609632
ap-southeast-2	798376113853

Region	ID account
ap-southeast-3	652364314486
ap-southeast-4	849137399833
ap-south-1	507069717855
ap-south-2	644042651268
ap-east-1	174803364771
sa-east-1	287228555773
me-south-1	855739686837
me-central-1	614065512851

Integrazioni fittizie per le API REST in API Gateway

Amazon API Gateway supporta le integrazioni fittizie per i metodi API. Questa caratteristica permette agli sviluppatori di API di generare risposte API da API Gateway direttamente, senza che sia necessario un back-end di integrazione. Uno sviluppatore di API può usare questa caratteristica per sbloccare team dipendenti che devono lavorare a un'API prima che lo sviluppo del progetto sia completato. Puoi anche usare questa caratteristica per allestire una pagina di destinazione per l'API, con una panoramica e comandi di navigazione per l'API. Per un esempio di pagina di destinazione di questo tipo, consulta la richiesta e la risposta di integrazione del metodo GET nella risorsa root dell'API di esempio illustrata in Tutorial: creazione di un'API REST mediante l'importazione di un esempio.

Uno sviluppatore di API può decidere in che modo API Gateway risponde alle richieste di integrazione fittizia. A tale scopo, è necessario configurare la richiesta e la risposta di integrazione del metodo per associare una risposta a un codice di stato specifico. Affinché un metodo con l'integrazione fittizia restituisca una risposta `200`, configura il modello di mappatura del corpo della richiesta di integrazione per restituire quanto segue.

```
{"statusCode": 200}
```

Configura una risposta di integrazione `200` con il modello di mappatura del corpo seguente, ad esempio:

```
{
    "statusCode": 200,
    "message": "Go ahead without me."
}
```

Analogamente, affinché il metodo restituisca, ad esempio, una risposta di errore `500`, configura il modello di mappatura del corpo della richiesta di integrazione per restituire quanto segue.

```
{"statusCode": 500}
```

Configura, ad esempio, una risposta di integrazione `500` con il modello di mappatura seguente:

```
{
    "statusCode": 500,
    "message": "The invoked method is not supported on the API resource."
}
```

In alternativa, puoi fare in modo che un metodo dell'integrazione fittizia restituisca la risposta di integrazione predefinita senza definire il modello di mappatura della richiesta di integrazione. La risposta di integrazione predefinita è quella con valore HTTP status regex (Regex stato HTTP) non definito. Assicurati che siano impostati i comportamenti di passthrough appropriati.

> **Note**
>
> Le integrazioni fittizie non sono concepite per supportare modelli di risposta di grandi dimensioni. Se ne hai bisogno per il caso d'uso specifico, dovresti valutare invece l'utilizzo di un'integrazione Lambda.

Usando un modello di mappatura della richiesta di integrazione puoi inserire la logica dell'applicazione per stabilire quale risposta di integrazione fittizia restituire in base a determinate condizioni. Puoi ad esempio usare un parametro di query `scope` nella richiesta in ingresso per determinare se restituire una risposta di esito positivo o di errore:

```
{
  #if( $input.params('scope') == "internal" )
```

```
    "statusCode": 200
  #else
    "statusCode": 500
  #end
}
```

In questo modo, il metodo dell'integrazione fittizia consente alle chiamate interne di passare, mentre gli altri tipi di chiamate vengono rifiutati con una risposta di errore.

Questa sezione descrive come utilizzare la console API Gateway per abilitare l'integrazione fittizia per un metodo API.

Argomenti

- Abilitazione dell'integrazione fittizia tramite la console API Gateway

Abilitazione dell'integrazione fittizia tramite la console API Gateway

In Gateway API deve essere disponibile un metodo. Segui le istruzioni in Tutorial: creare un file REST API con un'HTTPintegrazione non proxy.

1. Seleziona una risorsa API e scegli Crea metodo.

   Per creare il metodo, procedi nel seguente modo:

   a. Per Tipo di metodo seleziona un metodo.
   b. Per Tipo di integrazione seleziona Fittizio.
   c. Scegli Crea metodo.
   d. Nella scheda Richiesta metodo scegli Modifica in Impostazioni della richiesta del metodo.
   e. Scegli Parametri della stringa di query URL. Scegli Aggiungi stringa di query e per Nome immetti **`scope`**. Questo parametro di query determina se il chiamante è interno o meno.
   f. Selezionare Salva.
2. Nella scheda Risposta metodo scegli Crea risposta, quindi procedi come indicato di seguito:

   a. Per Stato HTTP immetti **`500`**.
   b. Selezionare Salva.
3. Nella scheda Richiesta di integrazione seleziona Modifica per Impostazioni della richiesta di integrazione.

4. Scegli Modelli di mappatura e procedi come indicato di seguito:

   a. Scegliere Add mapping template (Aggiungi modello di mappatura).

   b. Per Tipo di contenuto inserisci **`application/json`**.

   c. Per Corpo del modello inserisci quanto segue:

```
{
  #if( $input.params('scope') == "internal" )
    "statusCode": 200
  #else
    "statusCode": 500
  #end
}
```

   d. Selezionare Salva.

5. Nella scheda Risposta di integrazione scegli Modifica per Predefinito - Risposta.

6. Scegli Modelli di mappatura e procedi come indicato di seguito:

   a. Per Tipo di contenuto inserisci **`application/json`**.

   b. Per Corpo del modello inserisci quanto segue:

```
{
    "statusCode": 200,
    "message": "Go ahead without me"
}
```

   c. Selezionare Salva.

7. Scegli Crea risposta.

   Per creare una risposta 500, procedi come descritto qui di seguito:

   a. Per HTTP status regex (Regex stato HTTP), immettere **`5\d{2}`**.

   b. Per Stato metodo risposta seleziona **`500`**.

   c. Selezionare Salva.

   d. Per 5\d{2} - Risposta scegli Modifica.

   e. Scegli Modelli di mappatura, quindi seleziona Aggiungi modello di mappatura.

   f. Per Tipo di contenuto inserisci **`application/json`**.

   g. Per Corpo del modello inserisci quanto segue:

```
{
    "statusCode": 500,
    "message": "The invoked method is not supported on the API resource."
}
```

h. Selezionare Salva.

8. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda. Per testare l'integrazione fittizia, procedi come indicato di seguito:

   a. Immetti `scope=internal` in Stringhe di query. Scegli Test (Esegui test). Viene visualizzato il risultato del test:

```
Request: /?scope=internal
Status: 200
Latency: 26 ms
Response Body

{
  "statusCode": 200,
  "message": "Go ahead without me"
}

Response Headers

{"Content-Type":"application/json"}
```

   b. Immettere `scope=public` in `Query strings` oppure lasciare vuoto il campo. Scegli Test (Esegui test). Viene visualizzato il risultato del test:

```
Request: /
Status: 500
Latency: 16 ms
Response Body

{
  "statusCode": 500,
  "message": "The invoked method is not supported on the API resource."
}
```

```
Response Headers

{"Content-Type":"application/json"}
```

Puoi anche restituire le intestazioni in una risposta di integrazione fittizia aggiungendo prima di tutto un'intestazione a una risposta del metodo e quindi configurando una mappatura dell'intestazione nella risposta di integrazione. È in questo modo che la console API Gateway abilita il supporto CORS restituendo le intestazioni richieste da CORS.

Richiedi la convalida per REST APIs in API Gateway

È possibile configurare API Gateway per eseguire la convalida di base di una API richiesta prima di procedere con la richiesta di integrazione. Quando la convalida fallisce, API Gateway fallisce immediatamente la richiesta, restituisce una risposta di errore 400 al chiamante e pubblica i risultati della convalida in Logs. CloudWatch Questo comportamento riduce le chiamate non necessarie al back-end. Aspetto ancora più importante, ti permette di concentrarti sulle attività di convalida specifiche dell'applicazione. È possibile convalidare il corpo di una richiesta verificando che i parametri obbligatori della richiesta siano validi e diversi da null oppure specificando uno schema di modello per una convalida dei dati più complessa.

Argomenti

- Panoramica della convalida di base delle richieste in Gateway API
- Modelli di dati per REST APIs
- Imposta la convalida delle richieste di base in API Gateway
- AWS CloudFormation modello di un campione API con convalida della richiesta di base

Panoramica della convalida di base delle richieste in Gateway API

APIGateway può eseguire la convalida di base delle richieste, in modo che tu possa concentrarti sulla convalida specifica dell'app nel backend. Per la convalida, API Gateway verifica una o entrambe le seguenti condizioni:

- I parametri di richiesta richiesti nella URI stringa di query e nelle intestazioni di una richiesta in entrata sono inclusi e non sono vuoti.

- Il payload della richiesta applicabile aderisce alla richiesta JSONdello schema configurato del metodo.

Per attivare la convalida, si specificano le regole di convalida in un validatore di richiesta, si aggiunge il validatore alla mappa dei validatori di richiesta e si assegna il API validatore ai singoli metodi. API

> **Note**
>
> La convalida del corpo della richiesta e Comportamenti passthrough di integrazione sono due argomenti separati. Quando il payload di una richiesta non dispone di uno schema di modello corrispondente, puoi scegliere di eseguire il transito o il blocco del payload originale. Per ulteriori informazioni, consulta Comportamenti passthrough di integrazione.

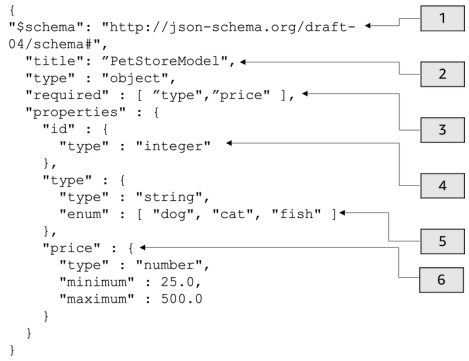
Modelli di dati per REST APIs

In API Gateway, un modello definisce la struttura dei dati di un payload. In API Gateway, i modelli vengono definiti utilizzando la bozza JSON dello schema 4. L'JSONoggetto seguente è costituito da dati di esempio nell'esempio di Pet Store.

```
{
    "id": 1,
    "type": "dog",
    "price": 249.99
}
```

I dati contengono i valori `id`, `type` e `price` dell'animale domestico. Un modello di questi dati consente di:

- Usare la convalida di base delle richieste.
- Creare modelli di mappatura per la trasformazione dei dati.
- Crea un tipo di dati definito dall'utente (UDT) quando generi unSDK.



In questo modello:

1. L'`$schema`oggetto rappresenta un identificatore di versione JSON dello schema valido. Questo schema è la bozza dello JSON schema v4.
2. L'oggetto `title` è un identificatore in formato leggibile del modello. Questo titolo è `PetStoreModel`.
3. La parola chiave di convalida `required` richiede `type` e `price` per la convalida di base della richiesta.
4. La variabile `properties` del modello è `id`, `type` e`price`. Ogni oggetto ha proprietà che vengono descritte nel modello.
5. L'oggetto `type` può avere solo i valori `dog`, `cat` o `fish`.
6. L'oggetto `price` è un numero ed è vincolato con un valore `minimum` di 25 e un valore `maximum` di 500.

PetStore modello

```
1 {
2 "$schema": "http://json-schema.org/draft-04/schema#",
3  "title": "PetStoreModel",
4  "type" : "object",
5  "required" : [ "price", "type" ],
6  "properties" : {
7    "id" : {
8      "type" : "integer"
9    },
10    "type" : {
11      "type" : "string",
12      "enum" : [ "dog", "cat", "fish" ]
13    },
```

```
14    "price" : {
15      "type" : "number",
16      "minimum" : 25.0,
17      "maximum" : 500.0
18    }
19  }
20 }
```

In questo modello:

1. Alla riga 2, l'`$schema`oggetto rappresenta un identificatore di versione JSON dello schema valido. Questo schema è la bozza dello JSON schema v4.
2. Alla riga 3, l'oggetto `title` è un identificatore in formato leggibile del modello. Questo titolo è `PetStoreModel`.
3. Alla riga 5, la parola chiave di convalida `required` richiede `type` e `price` per la convalida di base della richiesta.
4. Alle righe da 6 a 17, la variabile `properties` del modello è `id`. `type` e `price`. Ogni oggetto ha proprietà che vengono descritte nel modello.
5. Alla riga 12, l'oggetto `type` può avere solo i valori `dog`, `cat` o `fish`.
6. Alle righe da 14 a 17, l'oggetto `price` è un numero ed è vincolato con un valore `minimum` di 25 e un valore `maximum` di 500.

Creazione di modelli più complessi

È possibile utilizzare la primitiva `$ref` per creare definizioni riutilizzabili per modelli più lunghi. Ad esempio, è possibile creare una definizione chiamata `Price` nella sezione `definitions` che descrive l'oggetto `price`. Il valore di `$ref` è la definizione `Price`.

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "PetStoreModelReUsableRef",
  "required" : ["price", "type" ],
  "type" : "object",
  "properties" : {
    "id" : {
      "type" : "integer"
    },
    "type" : {
      "type" : "string",
```

```
      "enum" : [ "dog", "cat", "fish" ]
    },
    "price" : {
        "$ref": "#/definitions/Price"
    }
  },
  "definitions" : {
      "Price": {
        "type" : "number",
        "minimum" : 25.0,
        "maximum" : 500.0
            }
      }
}
```

È inoltre possibile fare riferimento a un altro schema del modello definito in un file di modello esterno. Impostare il valore della proprietà `$ref` sulla posizione del modello. Nell'esempio seguente, il `Price` modello è definito nel `PetStorePrice` modello in APIa1234.

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "PetStorePrice",
  "type": "number",
  "minimum": 25,
  "maximum": 500
}
```

Il modello più lungo può fare riferimento al modello `PetStorePrice`.

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "PetStoreModelReusableRefAPI",
  "required" : [ "price", "type" ],
  "type" : "object",
  "properties" : {
    "id" : {
      "type" : "integer"
    },
    "type" : {
      "type" : "string",
      "enum" : [ "dog", "cat", "fish" ]
    },
```

```
    "price" : {
        "$ref": "https://apigateway.amazonaws.com/restapis/a1234/models/PetStorePrice"
    }
  }
}
```

Utilizzo di modelli di dati di output

In caso di trasformazione dei dati, è possibile definire un modello di payload nella risposta dell'integrazione. Un modello di payload può essere utilizzato quando si genera unSDK. Per linguaggi fortemente tipizzati, come Java, Objective-C o Swift, l'oggetto corrisponde a un tipo di dati definito dall'utente (). UDT APIGateway crea un UDT file se gli fornisci un modello di dati quando generi un. SDK Per ulteriori informazioni sulle trasformazioni dei dati, consultare Modelli di mappatura per REST APIs.

L'esempio seguente mostra i dati in uscita da una risposta di integrazione.

```
{
[
  {
    "description" : "Item 1 is a dog.",
    "askingPrice" : 249.99
  },
  {
    "description" : "Item 2 is a cat.",
    "askingPrice" : 124.99
  },
  {
    "description" : "Item 3 is a fish.",
    "askingPrice" : 0.99
  }
]
}
```

L'esempio seguente è un modello di payload che descrive i dati di output.

```
{
"$schema": "http://json-schema.org/draft-04/schema#",
  "title": ”PetStoreOutputModel",
  "type" : "object",
  "required" : [ "description", "askingPrice" ],
  "properties" : {
```

```
    "description" : {
      "type" : "string"
    },
    "askingPrice" : {
      "type" : "number",
      "minimum" : 25.0,
      "maximum" : 500.0
    }
  }
}
```

Con questo modello, è possibile chiamare an SDK per recuperare i valori delle `askingPrice` proprietà `description` e leggendo le proprietà `PetStoreOutputModel[i].description` and`PetStoreOutputModel[i].askingPrice`. Se non viene fornito alcun modello, API Gateway utilizza il modello vuoto per creare un modello predefinitoUDT.

Passaggi successivi

- Questa sezione fornisce risorse che è possibile utilizzare per acquisire maggiori conoscenze sui concetti trattati in questo argomento.

  È possibile seguire i tutorial relativi alla convalida delle richieste:

  - Imposta la convalida delle richieste utilizzando la API console Gateway
  - Imposta la convalida delle richieste di base utilizzando il AWS CLI
  - Imposta la convalida delle richieste di base utilizzando una definizione aperta API
- È possibile ottenere ulteriori informazioni sulla trasformazione dei dati e sui modelli di mappatura, Modelli di mappatura per REST APIs.
- Puoi anche vedere modelli di dati più complicati. Per informazioni, consulta Esempi di modelli di dati e modelli di mappatura per Gateway API.

Imposta la convalida delle richieste di base in API Gateway

Questa sezione mostra come configurare la convalida delle richieste per API Gateway utilizzando la console e una definizione apertaAPI. AWS CLI

Argomenti

- Imposta la convalida delle richieste utilizzando la API console Gateway
- Imposta la convalida delle richieste di base utilizzando il AWS CLI

- Imposta la convalida delle richieste di base utilizzando una definizione aperta API

Imposta la convalida delle richieste utilizzando la API console Gateway

È possibile utilizzare la console API Gateway per convalidare una richiesta selezionando uno dei tre validatori per una richiesta: API

- Convalida del corpo.
- Convalida dei parametri e delle intestazioni delle stringhe di query.
- Convalida del corpo, dei parametri delle stringhe di query e delle intestazioni.

Quando applichi uno dei validatori a un API metodo, la console API Gateway aggiunge il validatore alla mappa. API RequestValidators

Per seguire questo tutorial, utilizzerai un AWS CloudFormation modello per creare un Gateway incompletoAPI. API Questo API ha una `/validator` risorsa con `GET` e `POST` metodi. Entrambi i metodi sono integrati con l'`http://petstore-demo-endpoint.execute-api.com/petstore/pets`HTTPendpoint. Configurare due tipi di convalida delle richieste:

- Nel `GET` metodo, configurerai la convalida della richiesta per i parametri della stringa di URL query.
- Nel metodo `POST`, verrà configurata la convalida delle richieste per il corpo della richiesta.

Ciò consentirà solo a determinate API chiamate di passare a. API

Scarica e decomprimi il modello di creazione dell'app per AWS CloudFormation. Utilizzerai questo modello per crearne uno incompletoAPI. Completerai il resto dei passaggi nella console API Gateway.

Per creare uno AWS CloudFormation stack

1. Apri la AWS CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformation.
2. Scegliere Create stack (Crea stack), quindi With new resources (standard) (Con nuove risorse (standard)).
3. In Specificare modello, scegliere Carica un file modello.
4. Selezionare il modello scaricato.

5. Seleziona Successivo.
6. Per Stack name (Nome stack), inserire **`request-validation-tutorial-console`**, quindi scegliere Next (Avanti).
7. Per Configure stack options (Configura opzioni di stack), scegliere Next (Successivo).
8. Per quanto riguarda le funzionalità, riconosci che AWS CloudFormation puoi creare IAM risorse nel tuo account.
9. Scegli Invia.

AWS CloudFormation fornisce le risorse specificate nel modello. Per completare il provisioning delle risorse, potrebbero essere necessari alcuni minuti. Quando lo stato del tuo AWS CloudFormation stack è CREATE_ COMPLETE, sei pronto per passare alla fase successiva.

Per selezionare il tuo appena creato API

1. Selezionare lo stack **`request-validation-tutorial-console`** appena creato.
2. Scegliere Resources (Risorse).
3. In Physical ID, scegli il tuoAPI. Questo collegamento ti indirizzerà alla console API Gateway.

Prima di modificare i metodi `GET` e `POST`, è necessario creare un modello.

Creazione di un modello

1. È necessario che un modello utilizzi la convalida della richiesta nel corpo di una richiesta in arrivo. Per creare un modello seleziona Modelli nel pannello di navigazione principale.
2. Scegli Crea modello.
3. Per Nome, immetti **`PetStoreModel`**.
4. In Tipo di contenuto, inserire **`application/json`**. Se non viene trovato alcun tipo di contenuto corrispondente, la convalida della richiesta non viene eseguita. Per utilizzare lo stesso modello indipendentemente dal tipo di contenuti, inserisci **`$default`**.
5. Per Descrizione immetti **`My PetStore Model`** come descrizione del modello.
6. Per Schema modello incolla il seguente modello nell'editor di codice e scegli Crea.

```
{
  "type" : "object",
  "required" : [ "name", "price", "type" ],
```

```
  "properties" : {
    "id" : {
      "type" : "integer"
    },
    "type" : {
      "type" : "string",
      "enum" : [ "dog", "cat", "fish" ]
    },
    "name" : {
      "type" : "string"
    },
    "price" : {
      "type" : "number",
      "minimum" : 25.0,
      "maximum" : 500.0
    }
  }
}
```

Per ulteriori informazioni sul modello, consulta Modelli di dati per REST APIs.

Per configurare la convalida della richiesta per un metodo **GET**

1. Nel riquadro di navigazione principale, scegli Risorse, quindi seleziona il GETmetodo.
2. Nella scheda Richiesta metodo, in Impostazioni richiesta metodo, scegli Modifica.
3. Per Validatore richiesta seleziona Convalida parametri di stringa query e intestazioni.
4. In Parametri della stringa di URL query, procedi come segue:
   a. Scegliere Add query string (Aggiungi stringa di query).
   b. Per Nome, immetti **petType**.
   c. Attiva Campo obbligatorio.
   d. Mantieni disattivata l'opzione Caching.
5. Selezionare Salva.
6. Nella scheda Richiesta di integrazione scegli Modifica in Impostazioni della richiesta di integrazione.
7. In Parametri della stringa di URL query, procedi come segue:
   a. Scegliere Add query string (Aggiungi stringa di query).

b. Per Nome, immetti **`petType`**.

c. In Mappato da, inserire **`method.request.querystring.petType`**. Questa operazione associa **`petType`** al tipo di animale domestico.

   Per ulteriori informazioni sulla mappatura dei dati, consultare il tutorial relativo alla mappatura dei dati.

d. Mantieni disattivata l'opzione Caching.

8. Selezionare Salva.

Per eseguire il test della convalida della richiesta per il metodo **`GET`**

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. In Stringhe di query immetti **`petType=dog`** e scegli Test.
3. Il test del metodo restituirà `200 OK` e un elenco di cani.

   Per informazioni su come trasformare questi dati di output, consultare il tutorial sulla mappatura dei dati.

4. Rimuovere **`petType=dog`** e scegliere Test.
5. Il test del metodo restituirà un errore `400` con il seguente messaggio di errore:

```
{
  "message": "Missing required request parameters: [petType]"
}
```

Per configurare la convalida della richiesta per il metodo **`POST`**

1. Nel riquadro di navigazione principale, scegliete Risorse, quindi selezionate il POSTmetodo.
2. Nella scheda Richiesta metodo, in Impostazioni richiesta metodo, scegli Modifica.
3. Per Validatore richiesta seleziona Convalida corpo.
4. In Corpo della richiesta scegli Aggiungi modello.
5. Per Tipo di contenuto**`application/json`**, immettete, quindi per Modello, selezionate PetStoreModel.
6. Selezionare Salva.

Per eseguire il test della convalida della richiesta per un metodo **POST**

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. In Corpo della richiesta incolla quanto segue nell'editor di codice:

```
{
  "id": 2,
  "name": "Bella",
  "type": "dog",
  "price": 400
}
```

   Scegli Test (Esegui test).
3. Il test del metodo restituirà `200 OK` e un messaggio di operazione riuscita.
4. In Corpo della richiesta incolla quanto segue nell'editor di codice:

```
{
  "id": 2,
  "name": "Bella",
  "type": "dog",
  "price": 4000
}
```

   Scegli Test (Esegui test).
5. Il test del metodo restituirà un errore `400` con il seguente messaggio di errore:

```
{
 "message": "Invalid request body"
}
```

   Nella parte inferiore dei log del test viene restituito il motivo del corpo della richiesta non valido. In questo caso, il prezzo dell'animale era maggiore del valore massimo specificato nel modello.

Per eliminare una AWS CloudFormation pila

1. Apri la AWS CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformation.

2. Seleziona il tuo AWS CloudFormation stack.
3. Scegli Elimina e conferma la tua scelta.

Passaggi successivi

- Per informazioni su come trasformare i dati di output ed eseguire ulteriori mappature dei dati, consultare il tutorial sulla mappatura dei dati.
- Eseguire il tutorial Configurazione della convalida di base delle richieste in AWS CLI per eseguire passaggi simili utilizzando la AWS CLI.

Imposta la convalida delle richieste di base utilizzando il AWS CLI

È possibile creare un validatore per configurare la convalida della richiesta utilizzando la AWS CLI. Per seguire questo tutorial, utilizzerai un AWS CloudFormation modello per creare un API Gateway API incompleto.

> **Note**
>
> Questo non è lo stesso AWS CloudFormation modello del tutorial per console.

Utilizzando una risorsa `/validator` precedentemente esposta, verranno creati i metodi `GET` e `POST`. Entrambi i metodi verranno integrati con l'`http://petstore-demo-endpoint.execute-api.com/petstore/pets`HTTPendpoint. Verranno configurate le due convalide delle richieste seguenti:

- Sul `GET` metodo, creerai un `params-only` validatore per convalidare i parametri della stringa di URL query.
- Nel metodo `POST`, verrà creato un validatore `body-only` per convalidare il corpo della richiesta.

Ciò consentirà solo a determinate API chiamate di passare a. API

Per creare una AWS CloudFormation pila

Scarica e decomprimi il modello di creazione dell'app per. AWS CloudFormation

Per completare il tutorial seguente, è necessario disporre della AWS Command Line Interface (AWS CLI) versione 2.

Per i comandi lunghi viene utilizzato un carattere di escape (\) per dividere un comando su più righe.

> **Note**
>
> In Windows, alcuni CLI comandi Bash che usi comunemente (come`zip`) non sono supportati dai terminali integrati del sistema operativo. Per ottenere una versione integrata su Windows di Ubuntu e Bash, installa il sottosistema Windows per Linux. CLII comandi di esempio in questa guida utilizzano la formattazione Linux. I comandi che includono JSON documenti in linea devono essere riformattati se si utilizza Windows. CLI

1. Usa il seguente comando per creare lo stack. AWS CloudFormation

```
aws cloudformation create-stack --stack-name request-validation-tutorial-cli
 --template-body file://request-validation-tutorial-cli.zip --capabilities
 CAPABILITY_NAMED_IAM
```

2. AWS CloudFormation fornisce le risorse specificate nel modello. Per completare il provisioning delle risorse, potrebbero essere necessari alcuni minuti. Usa il comando seguente per vedere lo stato del tuo AWS CloudFormation stack.

```
aws cloudformation describe-stacks --stack-name request-validation-tutorial-cli
```

3. Quando lo stato dello AWS CloudFormation stack è`StackStatus: "CREATE_COMPLETE"`, usa il seguente comando per recuperare i valori di output pertinenti per i passaggi futuri.

```
aws cloudformation describe-stacks --stack-name request-validation-tutorial-cli
--query "Stacks[*].Outputs[*].{OutputKey: OutputKey, OutputValue: OutputValue,
Description: Description}"
```

   Di seguito sono riportati i valori di output:

   - ApiId, che è l'ID per. API Per questo tutorial, l'APIID è`abc123`.
   - ResourceId, che è l'ID della risorsa di validazione in cui sono esposti `POST` i metodi `GET` and. Per questo tutorial, l'ID risorsa è `efg456`.

Creazione dei validatori della richiesta e importazione di un modello

1. Per utilizzare la convalida della richiesta con la AWS CLI, è necessario un validatore. Usare il seguente comando per creare un validatore che convalidi solo i parametri della richiesta.

```
aws apigateway create-request-validator --rest-api-id abc123 \
      --no-validate-request-body \
      --validate-request-parameters \
      --name params-only
```

   Annotare l'ID del validatore `params-only`.

2. Usare il seguente comando per creare un validatore che convalidi solo il corpo della richiesta.

```
aws apigateway create-request-validator --rest-api-id abc123 \
      --validate-request-body \
      --no-validate-request-parameters \
      --name body-only
```

   Annotare l'ID del validatore `body-only`.

3. È necessario che un modello utilizzi la convalida della richiesta nel corpo di una richiesta in arrivo. Utilizzare il comando seguente per importare un modello.

```
aws apigateway create-model --rest-api-id abc123 --name PetStoreModel --description
 'My PetStore Model' --content-type 'application/json' --schema '{"type":
 "object", "required" : [ "name", "price", "type" ], "properties" : { "id" :
 {"type" : "integer"},"type" : {"type" : "string", "enum" : [ "dog", "cat",
 "fish" ]},"name" : { "type" : "string"},"price" : {"type" : "number","minimum" :
 25.0, "maximum" : 500.0}}}}'
```

   Se non viene trovato alcun tipo di contenuto corrispondente, la convalida della richiesta non viene eseguita. Per utilizzare lo stesso modello indipendentemente dal tipo di contenuto, specifica `$default` come chiave.

Creazione dei metodi **`GET`** e **`POST`**

1. Utilizzate il comando seguente per aggiungere il `GET` HTTP metodo alla `/validate` risorsa. Questo comando crea il metodo `GET`, aggiunge il validatore `params-only` e imposta la stringa di query `petType` come richiesto.

```
aws apigateway put-method --rest-api-id abc123 \
       --resource-id efg456 \
       --http-method GET \
       --authorization-type "NONE" \
       --request-validator-id aaa111 \
       --request-parameters "method.request.querystring.petType=true"
```

Utilizzate il comando seguente per aggiungere il `POST` HTTP metodo alla `/validate` risorsa. Questo comando crea il metodo `POST`, aggiunge il validatore `body-only` e collega il modello al validatore solo del corpo.

```
aws apigateway put-method --rest-api-id abc123 \
       --resource-id efg456 \
       --http-method POST \
       --authorization-type "NONE" \
       --request-validator-id bbb222 \
       --request-models 'application/json'=PetStoreModel
```

2. Utilizzare il comando seguente per configurare la risposta `200 OK` del metodo `GET /validate`.

```
aws apigateway put-method-response --rest-api-id abc123  \
            --resource-id efg456 \
            --http-method GET \
            --status-code 200
```

Utilizzare il comando seguente per configurare la risposta `200 OK` del metodo `POST /validate`.

```
aws apigateway put-method-response --rest-api-id abc123  \
            --resource-id efg456 \
            --http-method POST \
            --status-code 200
```

3. Utilizzate il comando seguente per configurare un dispositivo `Integration` con un HTTP endpoint specificato per il `GET /validation` metodo.

```
aws apigateway put-integration --rest-api-id abc123  \
            --resource-id efg456 \
            --http-method GET \
            --type HTTP \
```

```
            --integration-http-method GET \
            --request-parameters '{"integration.request.querystring.type" :
 "method.request.querystring.petType"}' \
            --uri 'http://petstore-demo-endpoint.execute-api.com/petstore/pets'
```

Utilizzate il comando seguente per impostare un endpoint `Integration` con un HTTP endpoint specificato per il `POST /validation` metodo.

```
aws apigateway put-integration --rest-api-id abc123  \
              --resource-id efg456 \
              --http-method POST \
              --type HTTP \
              --integration-http-method GET \
              --uri 'http://petstore-demo-endpoint.execute-api.com/petstore/pets'
```

4. Utilizzare il comando seguente per configurare una risposta di integrazione per il metodo `GET /validation`.

```
aws apigateway put-integration-response --rest-api-id abc123 \
              --resource-id efg456\
              --http-method GET \
              --status-code 200 \
              --selection-pattern ""
```

Utilizzare il comando seguente per configurare una risposta di integrazione per il metodo `POST /validation`.

```
aws apigateway put-integration-response --rest-api-id abc123 \
            --resource-id efg456 \
            --http-method POST \
            --status-code 200 \
            --selection-pattern ""
```

Per testare il API

1. Per testare il metodo `GET`, che eseguirà la convalida della richiesta per le stringhe di query, usare il seguente comando:

```
aws apigateway test-invoke-method --rest-api-id abc123 \
            --resource-id efg456 \
```

```
            --http-method GET \
            --path-with-query-string '/validate?petType=dog'
```

Il risultato restituirà `200 OK` e l'elenco dei cani.

2. Utilizzare il seguente comando per eseguire il test senza includere la stringa di query `petType`.

```
aws apigateway test-invoke-method --rest-api-id abc123 \
            --resource-id efg456 \
            --http-method GET
```

Il risultato restituirà un errore `400`.

3. Per testare il metodo `POST`, che eseguirà la convalida della richiesta per il corpo della richiesta, usare il seguente comando:

```
aws apigateway test-invoke-method --rest-api-id abc123 \
            --resource-id efg456 \
            --http-method POST \
            --body '{"id": 1, "name": "bella", "type": "dog", "price" : 400 }'
```

Il risultato restituirà `200 OK` e un messaggio di operazione riuscita.

4. Usare il seguente comando per testare l'utilizzo di un corpo non valido.

```
aws apigateway test-invoke-method --rest-api-id abc123 \
             --resource-id efg456 \
             --http-method POST \
             --body '{"id": 1, "name": "bella", "type": "dog", "price" : 1000 }'
```

Il risultato restituirà un errore `400`, perché il prezzo del cane supera il prezzo massimo definito dal modello.

Per eliminare una AWS CloudFormation pila

- Usa il seguente comando per eliminare le tue AWS CloudFormation risorse.

```
aws cloudformation delete-stack  --stack-name request-validation-tutorial-cli
```

Imposta la convalida delle richieste di base utilizzando una definizione aperta API

È possibile dichiarare un validatore di richiesta a API livello specificando un insieme di x-amazon-apigateway-request-validatori. requestValidator oggetto oggetti nella x-amazon-apigateway-requestoggetto -validators mappa per selezionare quale parte della richiesta verrà convalidata. Nell'esempio Open API definition, ci sono due validatori:

- Il validatore `all`, che convalida sia il corpo, mediante il modello di dati `RequestBodyModel`, sia i parametri.

  Il modello di `RequestBodyModel` dati richiede che l'JSONoggetto di input contenga le `price` proprietà `nametype`, e. La proprietà `name` può essere qualsiasi stringa, `type` deve essere uno dei campi dell'enumerazione specificati (`["dog", "cat", "fish"]`) e `price` deve essere compreso tra 25 e 500. Il parametro `id` non è obbligatorio.
- Il validatore `param-only`, che convalida solo i parametri.

Per attivare un validatore di richiesta su tutti i metodi di anAPI, specificate una x-amazon-apigateway-requestproprietà -validator proprietà a API livello della API definizione Open. Nell'esempio Open API definition, il `all` validatore viene utilizzato su tutti i API metodi, a meno che non venga sovrascritto diversamente. Quando si utilizza un modello per convalidare il corpo, se non viene trovato alcun tipo di contenuto corrispondente, la convalida della richiesta non viene eseguita. Per utilizzare lo stesso modello indipendentemente dal tipo di contenuto, specifica `$default` come chiave.

Per attivare un validatore di richieste in un singolo metodo, specificare la proprietà `x-amazon-apigateway-request-validator` a livello di metodo. Nell'esempio, Open API definition, il `param-only` validatore sovrascrive il validatore sul `all` metodo. `GET`

Per importare l'APIesempio Open in API Gateway, consultate le seguenti istruzioni su o versoImportazione di un'API regionale in API Gateway. Importazione di un'API ottimizzata per l'edge in API Gateway

OpenAPI 3.0

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "ReqValidators Sample",
    "version" : "1.0.0"
  },
```

```
  "servers" : [ {
    "url" : "/{basePath}",
    "variables" : {
      "basePath" : {
        "default" : "/v1"
      }
    }
  } ],
  "paths" : {
    "/validation" : {
      "get" : {
        "parameters" : [ {
          "name" : "q1",
          "in" : "query",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "headers" : {
              "test-method-response-header" : {
                "schema" : {
                  "type" : "string"
                }
              }
            },
            "content" : {
              "application/json" : {
                "schema" : {
                  "$ref" : "#/components/schemas/ArrayOfError"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-request-validator" : "params-only",
        "x-amazon-apigateway-integration" : {
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
          "responses" : {
            "default" : {
```

```
              "statusCode" : "400",
              "responseParameters" : {
                "method.response.header.test-method-response-header" : "'static
value'"
              },
              "responseTemplates" : {
                "application/xml" : "xml 400 response template",
                "application/json" : "json 400 response template"
              }
            },
            "2\\d{2}" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.querystring.type" : "method.request.querystring.q1"
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "http"
        }
      },
      "post" : {
        "parameters" : [ {
          "name" : "h1",
          "in" : "header",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        } ],
        "requestBody" : {
          "content" : {
            "application/json" : {
              "schema" : {
                "$ref" : "#/components/schemas/RequestBodyModel"
              }
            }
          },
          "required" : true
        },
        "responses" : {
          "200" : {
            "description" : "200 response",
            "headers" : {
```

```
              "test-method-response-header" : {
                "schema" : {
                  "type" : "string"
                }
              }
            },
            "content" : {
              "application/json" : {
                "schema" : {
                  "$ref" : "#/components/schemas/ArrayOfError"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-request-validator" : "all",
        "x-amazon-apigateway-integration" : {
          "httpMethod" : "POST",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
          "responses" : {
            "default" : {
              "statusCode" : "400",
              "responseParameters" : {
                "method.response.header.test-method-response-header" : "'static
value'"
              },
              "responseTemplates" : {
                "application/xml" : "xml 400 response template",
                "application/json" : "json 400 response template"
              }
            },
            "2\\d{2}" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.header.custom_h1" : "method.request.header.h1"
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "http"
        }
      }
    }
  },
```

```
  "components" : {
    "schemas" : {
      "RequestBodyModel" : {
        "required" : [ "name", "price", "type" ],
        "type" : "object",
        "properties" : {
          "id" : {
            "type" : "integer"
          },
          "type" : {
            "type" : "string",
            "enum" : [ "dog", "cat", "fish" ]
          },
          "name" : {
            "type" : "string"
          },
          "price" : {
            "maximum" : 500.0,
            "minimum" : 25.0,
            "type" : "number"
          }
        }
      },
      "ArrayOfError" : {
        "type" : "array",
        "items" : {
          "$ref" : "#/components/schemas/Error"
        }
      },
      "Error" : {
        "type" : "object"
      }
    }
  },
  "x-amazon-apigateway-request-validators" : {
    "all" : {
      "validateRequestParameters" : true,
      "validateRequestBody" : true
    },
    "params-only" : {
      "validateRequestParameters" : true,
      "validateRequestBody" : false
    }
  }
```

```
}
```

OpenAPI 2.0

```
{
  "swagger" : "2.0",
  "info" : {
    "version" : "1.0.0",
    "title" : "ReqValidators Sample"
  },
  "basePath" : "/v1",
  "schemes" : [ "https" ],
  "paths" : {
    "/validation" : {
      "get" : {
        "produces" : [ "application/json", "application/xml" ],
        "parameters" : [ {
          "name" : "q1",
          "in" : "query",
          "required" : true,
          "type" : "string"
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "schema" : {
              "$ref" : "#/definitions/ArrayOfError"
            },
            "headers" : {
              "test-method-response-header" : {
                "type" : "string"
              }
            }
          }
        },
        "x-amazon-apigateway-request-validator" : "params-only",
        "x-amazon-apigateway-integration" : {
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
          "responses" : {
            "default" : {
              "statusCode" : "400",
              "responseParameters" : {
```

```
                "method.response.header.test-method-response-header" : "'static
value'"
              },
              "responseTemplates" : {
                "application/xml" : "xml 400 response template",
                "application/json" : "json 400 response template"
              }
            },
            "2\\d{2}" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.querystring.type" : "method.request.querystring.q1"
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "http"
        }
      },
      "post" : {
        "consumes" : [ "application/json" ],
        "produces" : [ "application/json", "application/xml" ],
        "parameters" : [ {
          "name" : "h1",
          "in" : "header",
          "required" : true,
          "type" : "string"
        }, {
          "in" : "body",
          "name" : "RequestBodyModel",
          "required" : true,
          "schema" : {
            "$ref" : "#/definitions/RequestBodyModel"
          }
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "schema" : {
              "$ref" : "#/definitions/ArrayOfError"
            },
            "headers" : {
              "test-method-response-header" : {
                "type" : "string"
```

```
              }
            }
          }
        },
        "x-amazon-apigateway-request-validator" : "all",
        "x-amazon-apigateway-integration" : {
          "httpMethod" : "POST",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
          "responses" : {
            "default" : {
              "statusCode" : "400",
              "responseParameters" : {
                "method.response.header.test-method-response-header" : "'static
value'"
              },
              "responseTemplates" : {
                "application/xml" : "xml 400 response template",
                "application/json" : "json 400 response template"
              }
            },
            "2\\d{2}" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.header.custom_h1" : "method.request.header.h1"
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "http"
        }
      }
    }
  },
  "definitions" : {
    "RequestBodyModel" : {
      "type" : "object",
      "required" : [ "name", "price", "type" ],
      "properties" : {
        "id" : {
          "type" : "integer"
        },
        "type" : {
          "type" : "string",
          "enum" : [ "dog", "cat", "fish" ]
```

```
        },
        "name" : {
          "type" : "string"
        },
        "price" : {
          "type" : "number",
          "minimum" : 25.0,
          "maximum" : 500.0
        }
      }
    },
    "ArrayOfError" : {
      "type" : "array",
      "items" : {
        "$ref" : "#/definitions/Error"
      }
    },
    "Error" : {
      "type" : "object"
    }
  },
  "x-amazon-apigateway-request-validators" : {
    "all" : {
      "validateRequestParameters" : true,
      "validateRequestBody" : true
    },
    "params-only" : {
      "validateRequestParameters" : true,
      "validateRequestBody" : false
    }
  }
}
```

AWS CloudFormation modello di un campione API con convalida della richiesta di base

Il seguente AWS CloudFormation esempio di definizione del modello definisce un esempio API con la convalida della richiesta abilitata. APIÈ un sottoinsieme di. PetStoreAPI L'API espone un metodo `POST` per aggiungere un animale domestico alla raccolta `pets` e un metodo `GET` per l'esecuzione di query relative agli animali domestici in base a un tipo specificato.

Sono dichiarati due validatori di richiesta:

GETValidator

Questo validatore è abilitato nel metodo `GET`. Consente a API Gateway di verificare che il parametro di query richiesto (`q1`) sia incluso e non vuoto nella richiesta in entrata.

POSTValidator

Questo validatore è abilitato nel metodo `POST`. Consente a API Gateway di verificare che il formato della richiesta di payload sia conforme a quello specificato`RequestBodyModel`. `application/json` Se non viene trovato alcun tipo di contenuto corrispondente, la convalida della richiesta non viene eseguita. Per utilizzare lo stesso modello indipendentemente dal tipo di contenuto, specifica `$default`. `RequestBodyModel` contiene un modello aggiuntivo, `RequestBodyModelId`, per definire l'ID dell'animale domestico.

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  StageName:
    Type: String
    Default: v1
    Description: Name of API stage.
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Name: ReqValidatorsSample
  RequestBodyModelId:
    Type: 'AWS::ApiGateway::Model'
    Properties:
      RestApiId: !Ref Api
      ContentType: application/json
      Description: Request body model for Pet ID.
      Schema:
        $schema: 'http://json-schema.org/draft-04/schema#'
        title: RequestBodyModelId
        properties:
            id:
              type: integer
  RequestBodyModel:
    Type: 'AWS::ApiGateway::Model'
    Properties:
      RestApiId: !Ref Api
      ContentType: application/json
```

```
      Description: Request body model for Pet type, name, price, and ID.
      Schema:
        $schema: 'http://json-schema.org/draft-04/schema#'
        title: RequestBodyModel
        required:
          - price
          - name
          - type
        type: object
        properties:
            id:
              "$ref": !Sub
                - 'https://apigateway.amazonaws.com/restapis/${Api}/models/
${RequestBodyModelId}'
                - Api: !Ref Api
                  RequestBodyModelId: !Ref RequestBodyModelId
            price:
              type: number
              minimum: 25
              maximum: 500
            name:
              type: string
            type:
              type: string
              enum:
                - "dog"
                - "cat"
                - "fish"
  GETValidator:
    Type: AWS::ApiGateway::RequestValidator
    Properties:
      Name: params-only
      RestApiId: !Ref Api
      ValidateRequestBody: False
      ValidateRequestParameters: True
  POSTValidator:
    Type: AWS::ApiGateway::RequestValidator
    Properties:
      Name: body-only
      RestApiId: !Ref Api
      ValidateRequestBody: True
      ValidateRequestParameters: False
  ValidationResource:
    Type: 'AWS::ApiGateway::Resource'
```

```
    Properties:
      RestApiId: !Ref Api
      ParentId: !GetAtt Api.RootResourceId
      PathPart: 'validation'
  ValidationMethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref ValidationResource
      HttpMethod: GET
      AuthorizationType: NONE
      RequestValidatorId: !Ref GETValidator
      RequestParameters:
        method.request.querystring.q1: true
      Integration:
        Type: HTTP_PROXY
        IntegrationHttpMethod: GET
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
  ValidationMethodPost:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref ValidationResource
      HttpMethod: POST
      AuthorizationType: NONE
      RequestValidatorId: !Ref POSTValidator
      RequestModels:
        application/json : !Ref RequestBodyModel
      Integration:
        Type: HTTP_PROXY
        IntegrationHttpMethod: POST
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
  ApiDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn:
      - ValidationMethodGet
      - RequestBodyModel
    Properties:
      RestApiId: !Ref Api
      StageName: !Sub '${StageName}'
Outputs:
  ApiRootUrl:
    Description: Root Url of the API
    Value: !Sub 'https://${Api}.execute-api.${AWS::Region}.amazonaws.com/${StageName}'
```

Trasformazioni dei dati per REST APIs in Gateway API

In API Gateway, una richiesta API di metodo può accettare un payload in un formato diverso dal payload della richiesta di integrazione. Analogamente, il back-end potrebbe restituire un payload delle risposta di integrazione diverso dal payload della risposta di metodo. È possibile mappare i parametri del URL percorso, i parametri della stringa di URL query, le HTTP intestazioni e il corpo della richiesta su API Gateway utilizzando modelli di mappatura.

Un modello di mappatura è uno script espresso in Velocity Template Language (VTL) e applicato al payload utilizzando espressioni. JSONPath

Il payload può avere un modello di dati secondo la bozza dello JSON schema 4. Per ulteriori informazioni sui modelli, consultare Modelli di dati per REST APIs.

> **Note**
>
> Non è necessario definire alcun modello per creare un modello di mappatura, ma è necessario definire un modello per consentire a API Gateway di generare SDK o attivare la convalida del corpo della richiesta per il proprio. API

Argomenti

- Modelli di mappatura per REST APIs
- Configura le trasformazioni dei dati in API Gateway
- Utilizza un modello di mappatura per sovrascrivere i parametri API di richiesta e risposta e i codici di stato di an
- Imposta le mappature dei dati di richiesta e risposta utilizzando la console Gateway API
- Esempi di modelli di dati e modelli di mappatura per Gateway API
- Riferimento alla mappatura dei dati di API richiesta e risposta di Amazon API Gateway
- APIModello di mappatura del gateway e riferimento alla variabile di registrazione degli accessi

Modelli di mappatura per REST APIs

In API Gateway, API la richiesta o la risposta di un metodo può richiedere un payload in un formato diverso dalla richiesta o dalla risposta di integrazione.

È possibile trasformare i tuoi dati con le seguenti finalità:

- Abbina il payload a un formato specificatoAPI.
- Sostituisci i parametri API di richiesta e risposta e i codici di stato di an.
- Restituire le intestazioni di risposta selezionate dal client.
- Associa i parametri di percorso, i parametri della stringa di query o i parametri di intestazione nella richiesta del metodo del HTTP proxy o Servizio AWS del proxy.
- Seleziona i dati da inviare utilizzando l'integrazione Servizi AWS, ad esempio le funzioni o gli endpoint di Amazon DynamoDB o Lambda. HTTP

È possibile usare modelli di mappatura per trasformare i dati. Un modello di mappatura è uno script espresso in Velocity Template Language (VTL) e applicato al payload utilizzando. JSONPath

Questa sezione descrive le informazioni concettuali relative ai modelli di mappatura. Per istruzioni sulla creazione di un modello di mappatura per un API Gateway RESTAPI, vedere. the section called “Configura le trasformazioni dei dati in API Gateway”

Esempio di modello di mappatura

L'esempio seguente è l'input dei dati in una richiesta di integrazione.

```
[
  {
    "id": 1,
    "type": "dog",
    "price": 249.99
  },
  {
    "id": 2,
    "type": "cat",
    "price": 124.99
  },
  {
    "id": 3,
    "type": "fish",
    "price": 0.99
  }
]
```

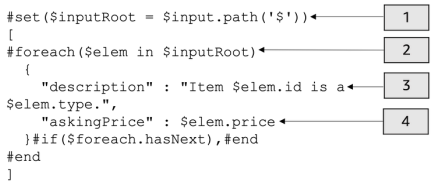
L'esempio seguente è un modello di mappatura per trasformare i dati della richiesta di integrazione.

```
#set($inputRoot = $input.path('$'))
[
#foreach($elem in $inputRoot)
  {
    "description" : "Item $elem.id is a $elem.type.",
    "askingPrice" : $elem.price
  }#if($foreach.hasNext),#end

#end
]
```

L'esempio seguente mostra i dati di output della trasformazione.

```
[
  {
    "description" : "Item 1 is a dog.",
    "askingPrice" : 249.99
  },
  {
    "description" : "Item 2 is a cat.",
    "askingPrice" : 124.99
  },
  {
    "description" : "Item 3 is a fish.",
    "askingPrice" : 0.99
  }
]
```

Il diagramma seguente mostra i dettagli di questo modello di mappatura.



1. La `$inputRoot` variabile rappresenta l'oggetto principale nei JSON dati originali della sezione precedente. Le direttive iniziano con il simbolo #.
2. Un `foreach` ciclo scorre attraverso ogni oggetto nei JSON dati originali.
3. La descrizione è una concatenazione dei dati del Pet `id` e `type` dei dati originali. JSON
4. `askingPricepricee` il prezzo indicato nei dati originali. JSON

PetStore modello di mappatura

```
1 #set($inputRoot = $input.path('$'))
2 [
3 #foreach($elem in $inputRoot)
4  {
5    "description" : "Item $elem.id is a $elem.type.",
6    "askingPrice" : $elem.price
7  }#if($foreach.hasNext),#end
8 #end
9 ]
```

In questo modello di mappatura:

1. Nella riga 1, la `$inputRoot` variabile rappresenta l'oggetto principale nei JSON dati originali della sezione precedente. Le direttive iniziano con il simbolo #.
2. Alla riga 3, un `foreach` ciclo scorre attraverso ogni oggetto nei JSON dati originali.
3. Alla riga 5, `description` c'è una concatenazione dei dati Pet `id` e dei dati originali`type`. JSON
4. Alla riga 6, `askingPrice` `price` è il prezzo in base ai dati originali. JSON

Per ulteriori informazioni sul Velocity Template Language, consulta Apache Velocity - Reference. VTL Per ulteriori informazioni suJSONPath, vedere JSONPath - for. XPath JSON

Il modello di mappatura presuppone che i dati sottostanti appartengano a un JSON oggetto. Non è richiesta la definizione di un modello per i dati. Tuttavia, un modello per i dati di output consente di restituire i dati precedenti come oggetto specifico del linguaggio. Per ulteriori informazioni, consulta Modelli di dati per REST APIs.

Esempi di modelli di mappatura complessi

Puoi anche creare modelli di mappatura più complessi. L'esempio seguente mostra la concatenazione dei riferimenti e il limite di 100 per determinare se un animale domestico è conveniente.

L'esempio seguente fornisce dati di input per una richiesta di integrazione.

```
[
  {
    "id": 1,
    "type": "dog",
```

```
    "price": 249.99
  },
  {
    "id": 2,
    "type": "cat",
    "price": 124.99
  },
  {
    "id": 3,
    "type": "fish",
    "price": 0.99
  }
]
```

L'esempio seguente è un modello di mappatura per trasformare i dati della richiesta di integrazione.

```
#set($inputRoot = $input.path('$'))
#set($cheap = 100)
[
#foreach($elem in $inputRoot)
  {
#set($name = "${elem.type}number$elem.id")
    "name" : $name,
    "description" : "Item $elem.id is a $elem.type.",
    #if($elem.price > $cheap )#set ($afford = 'too much!') #{else}#set ($afford =
 $elem.price)#end
"askingPrice" : $afford
  }#if($foreach.hasNext),#end

#end
]
```

L'esempio seguente mostra i dati di output della trasformazione.

```
[
  {
    "name" : dognumber1,
    "description" : "Item 1 is a dog.",
    "askingPrice" : too much!
  },
  {
    "name" : catnumber2,
    "description" : "Item 2 is a cat.",
```

```
    "askingPrice" : too much!
  },
  {
    "name" : fishnumber3,
    "description" : "Item 3 is a fish.",
    "askingPrice" : 0.99
  }
]
```

Puoi anche vedere modelli di dati più complicati. Per informazioni, consulta Esempi di modelli di dati e modelli di mappatura per Gateway API.

Configura le trasformazioni dei dati in API Gateway

Questa sezione mostra come configurare modelli di mappatura per trasformare le richieste e le risposte di integrazione utilizzando la console e AWS CLI.

Argomenti

- Configurare la trasformazione dei dati utilizzando la console API Gateway
- Imposta la trasformazione dei dati utilizzando il AWS CLI
- AWS CloudFormation Modello di trasformazione dei dati completato
- Passaggi successivi

Configurare la trasformazione dei dati utilizzando la console API Gateway

In questo tutorial, creerai una tabella DynamoDB incompleta API utilizzando il seguente file.zip .zip. data-transformation-tutorial-console Questo file incompleto API ha una `/pets` risorsa con e metodi. `GET` `POST`

- Il `GET` metodo otterrà i dati dall'`http://petstore-demo-endpoint.execute-api.com/petstore/pets`HTTPendpoint. I dati di output verranno trasformati in base al modello di mappatura in PetStore modello di mappatura.
- Il metodo `POST` consentirà all'utente di pubblicare (`POST`) le informazioni sugli animali domestici in una tabella Amazon DynamoDB utilizzando un modello di mappatura.

Scarica e decomprimi il modello di creazione dell'app per. AWS CloudFormation Utilizzerai questo modello per creare una tabella DynamoDB per pubblicare informazioni sugli animali domestici e una tabella incompleta. API Completerai il resto dei passaggi nella console API Gateway.

Per creare una AWS CloudFormation pila

1. Apri la AWS CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformation.
2. Scegliere Create stack (Crea stack), quindi With new resources (standard) (Con nuove risorse (standard)).
3. In Specificare modello, scegliere Carica un file modello.
4. Selezionare il modello scaricato.
5. Seleziona Successivo.
6. Per Stack name (Nome stack), inserire **`data-transformation-tutorial-console`**, quindi scegliere Next (Avanti).
7. Per Configure stack options (Configura opzioni di stack), scegliere Next (Successivo).
8. Per quanto riguarda le funzionalità, riconosci che AWS CloudFormation puoi creare IAM risorse nel tuo account.
9. Scegli Invia.

AWS CloudFormation fornisce le risorse specificate nel modello. Per completare il provisioning delle risorse, potrebbero essere necessari alcuni minuti. Quando lo stato del tuo AWS CloudFormation stack è CREATE_ COMPLETE, sei pronto per passare alla fase successiva.

Test della risposta di integrazione **`GET`**

1. Nella scheda Risorse dello AWS CloudFormation stack di**`data-transformation-tutorial-console`**, seleziona l'ID fisico del tuo. API
2. Nel riquadro di navigazione principale, scegli Risorse, quindi seleziona il GETmetodo.
3. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.

   L'output del test mostrerà quanto segue:

```
[
  {
    "id": 1,
    "type": "dog",
    "price": 249.99
  },
  {
```

```
    "id": 2,
    "type": "cat",
    "price": 124.99
  },
  {
    "id": 3,
    "type": "fish",
    "price": 0.99
  }
]
```

Questo output verrà trasformato in base al modello di mappatura in PetStore modello di mappatura.

Trasformazione della risposta di integrazione **GET**

1. Scegli la scheda Risposta di integrazione.

   Attualmente non sono definiti modelli di mappatura, quindi la risposta di integrazione non verrà trasformata.
2. Per Predefinito - Risposta scegli Modifica.
3. Scegli Modelli di mappatura e procedi come indicato di seguito:

   a. Scegliere Add mapping template (Aggiungi modello di mappatura).

   b. Per Tipo di contenuto inserisci **application/json**.

   c. Per Corpo del modello inserisci quanto segue:

```
#set($inputRoot = $input.path('$'))
[
#foreach($elem in $inputRoot)
  {
    "description" : "Item $elem.id is a $elem.type.",
    "askingPrice" : $elem.price
  }#if($foreach.hasNext),#end

#end
]
```

   Selezionare Salva.

Test della risposta di integrazione **GET**

- Scegli la scheda Test, quindi seleziona Test.

  L'output del test mostrerà la risposta trasformata.

```
[
  {
    "description" : "Item 1 is a dog.",
    "askingPrice" : 249.99
  },
  {
    "description" : "Item 2 is a cat.",
    "askingPrice" : 124.99
  },
  {
    "description" : "Item 3 is a fish.",
    "askingPrice" : 0.99
  }
]
```

Trasformazione dei dati di input dal metodo **POST**

1. Scegli il metodo POST.
2. Scegli la scheda Richiesta di integrazione, quindi seleziona Modifica per Impostazioni della richiesta di integrazione.

   Il AWS CloudFormation modello ha compilato alcuni campi di richiesta di integrazione.

   - Il tipo di integrazione è Servizio AWS.
   - Servizio AWS Si tratta di DynamoDB.
   - Il HTTP metodo è. `POST`
   - L'operazione è `PutItem`.
   - Il ruolo di esecuzione che consente a API Gateway di inserire un elemento nella tabella DynamoDB è. `data-transformation-tutorial-console-APIGatewayRole` AWS CloudFormation ha creato questo ruolo per consentire a API Gateway di disporre delle autorizzazioni minime per interagire con DynamoDB.

Il nome della tabella DynamoDB non è stato specificato. Il nome verrà specificato nei passaggi seguenti.

3. Per Richiesta corpo passthrough seleziona Mai.

   Ciò significa che API rifiuteranno i dati con Content-Types che non dispongono di un modello di mappatura.

4. Scegli Modelli di mappatura.
5. Tipo di contenuto è impostato su `application/json`. Ciò significa che i tipi di contenuto che non sono application/json verranno rifiutati da. API Per ulteriori informazioni sui comportamenti passthrough di integrazione, consultare Comportamenti passthrough di integrazione.
6. Inserire il codice seguente nell'editor di testo.

```
{
    "TableName":"data-transformation-tutorial-console-ddb",
    "Item": {
        "id": {
            "N": $input.json("$.id")
        },
        "type": {
            "S": $input.json("$.type")
        },
        "price": {
            "N": $input.json("$.price")
        }
    }
}
```

   Questo modello specifica la tabella come `data-transformation-tutorial-console-ddb` e imposta gli elementi come `id`, `type` e`price`. Gli elementi proverranno dal corpo del metodo `POST`. È anche possibile utilizzare un modello di dati per creare un modello di mappatura. Per ulteriori informazioni, consulta Richiedi la convalida per REST APIs in API Gateway.

7. Scegliere il pulsante Salva per salvare il modello di mappatura.

Aggiunta di un metodo e una risposta di integrazione dal metodo **POST**

Hanno AWS CloudFormation creato un metodo vuoto e una risposta di integrazione. Questa risposta verrà modificata per fornire ulteriori informazioni. Per ulteriori informazioni su come modificare le risposte, consultare Riferimento alla mappatura dei dati di API richiesta e risposta di Amazon API Gateway.

1. Nella scheda Risposta di integrazione scegli Modifica per Predefinito - Risposta.
2. Scegli Modelli di mappatura, quindi seleziona Aggiungi modello di mappatura.
3. Per Content-type immetti **application/json**.
4. Nell'editor di codice, inserire il seguente modello di mappatura di output per inviare un messaggio di output:

   ```
   { "message" : "Your response was recorded at $context.requestTime" }
   ```

   Per ulteriori informazioni sulle variabili di contesto, consultare `$context`Variabili per modelli di dati, autorizzatori, modelli di mappatura e registrazione degli accessi CloudWatch .
5. Scegliere il pulsante Salva per salvare il modello di mappatura.

Test del metodo **POST**

Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.

1. Nel corpo della richiesta, inserire il seguente esempio.

   ```
   {
             "id": "4",
             "type" : "dog",
             "price": "321"
   }
   ```
2. Scegli Test (Esegui test).

   L'output dovrebbe mostrare il messaggio di operazione riuscita.

   Puoi aprire la console DynamoDB https://console.aws.amazon.com/dynamodb/all'indirizzo per verificare che l'elemento di esempio sia nella tua tabella.

Per eliminare uno stack AWS CloudFormation

1. Apri la AWS CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformation.
2. Seleziona il tuo AWS CloudFormation stack.
3. Scegli Elimina e conferma la tua scelta.

Imposta la trasformazione dei dati utilizzando il AWS CLI

In questo tutorial, creerai una tabella DynamoDB incompleta API utilizzando il seguente file.zip .zip. data-transformation-tutorial-cli Questa incompleta API ha una `/pets` risorsa con un `GET` metodo integrato con l'endpoint. `http://petstore-demo-endpoint.execute-api.com/petstore/pets` HTTP Verrà creato un metodo `POST` per connettersi a una tabella DynamoDB e si useranno modelli di mappatura per inserire dati in una tabella DynamoDB.

- I dati di output verranno quindi trasformati in base al modello di mappatura in PetStore modello di mappatura.
- Verrà creato un metodo `POST` per consentire all'utente di pubblicare (`POST`) le informazioni sugli animali domestici in una tabella Amazon DynamoDB utilizzando un modello di mappatura.

Per creare una pila AWS CloudFormation

Scarica e decomprimi il modello di creazione dell'app per. AWS CloudFormation

Per completare il tutorial seguente, è necessario disporre della AWS Command Line Interface (AWS CLI) versione 2.

Per i comandi lunghi viene utilizzato un carattere di escape (\) per dividere un comando su più righe.

> **Note**
>
> In Windows, alcuni CLI comandi Bash che usi comunemente (come`zip`) non sono supportati dai terminali integrati del sistema operativo. Per ottenere una versione integrata su Windows di Ubuntu e Bash, installa il sottosistema Windows per Linux. CLII comandi di esempio in questa guida utilizzano la formattazione Linux. I comandi che includono JSON documenti in linea devono essere riformattati se si utilizza Windows. CLI

1. Usa il seguente comando per creare lo stack. AWS CloudFormation

```
aws cloudformation create-stack --stack-name data-transformation-tutorial-cli
 --template-body file://data-transformation-tutorial-cli.zip --capabilities
 CAPABILITY_NAMED_IAM
```

2. AWS CloudFormation fornisce le risorse specificate nel modello. Per completare il provisioning delle risorse, potrebbero essere necessari alcuni minuti. Usa il comando seguente per vedere lo stato del tuo AWS CloudFormation stack.

```
aws cloudformation describe-stacks --stack-name data-transformation-tutorial-cli
```

3. Quando lo stato dello AWS CloudFormation stack è`StackStatus: "CREATE_COMPLETE"`, usa il seguente comando per recuperare i valori di output pertinenti per i passaggi futuri.

```
 aws cloudformation describe-stacks --stack-name data-transformation-tutorial-cli
 --query "Stacks[*].Outputs[*].{OutputKey: OutputKey, OutputValue: OutputValue,
 Description: Description}"
```

   Di seguito sono riportati i valori di output:

   - ApiRole, che è il nome del ruolo che consente a API Gateway di inserire elementi nella tabella DynamoDB. Per questo tutorial, il nome del ruolo è `data-transformation-tutorial-cli-APIGatewayRole-ABCDEFG`.
   - DDBTableName, che è il nome della tabella DynamoDB. Per questo tutorial il nome della tabella è `data-transformation-tutorial-cli-ddb`.
   - ResourceId, che è l'ID della risorsa pets in cui sono esposti i `POST` metodi `GET` and. Per questo tutorial, l'ID risorsa è `efg456`.
   - ApiId, che è l'ID perAPI. Per questo tutorial, l'APIID è`abc123`.

Test del metodo **`GET`** prima della trasformazione dei dati

- Utilizzare il seguente comando per eseguire il test del metodo `GET`.

```
aws apigateway test-invoke-method --rest-api-id abc123 \
          --resource-id efg456 \
          --http-method GET
```

L'output del test riporterà le seguenti informazioni.

```
[
  {
    "id": 1,
    "type": "dog",
    "price": 249.99
  },
  {
    "id": 2,
    "type": "cat",
    "price": 124.99
  },
  {
    "id": 3,
    "type": "fish",
    "price": 0.99
  }
]
```

Questo output verrà trasformato in base al modello di mappatura in PetStore modello di mappatura.

Trasformazione della risposta di integrazione **GET**

- Utilizzare il comando seguente per aggiornare una risposta di integrazione per il metodo `GET`. Sostituisci il *`rest-api-id`* e *`resource-id`* con i tuoi valori.

  Per creare una risposta di integrazione, utilizzare il comando seguente.

```
aws apigateway put-integration-response --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method GET \
  --status-code 200 \
  --selection-pattern "" \
  --response-templates '{"application/json": "#set($inputRoot = $input.path(\"$
\"))\n[\n#foreach($elem in $inputRoot)\n {\n  \"description\": \"Item $elem.id is a
 $elem.type\",\n  \"askingPrice\": \"$elem.price\"\n }#if($foreach.hasNext),#end\n
\n#end\n]"}'
```

Test del metodo **GET**

- Utilizzare il seguente comando per eseguire il test del metodo `GET`.

```
aws apigateway test-invoke-method --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method GET \
```

L'output del test mostrerà la risposta trasformata.

```
[
  {
    "description" : "Item 1 is a dog.",
    "askingPrice" : 249.99
  },
  {
    "description" : "Item 2 is a cat.",
    "askingPrice" : 124.99
  },
  {
    "description" : "Item 3 is a fish.",
    "askingPrice" : 0.99
  }
]
```

Creazione di un metodo **POST**

1. Utilizzare il seguente comando per creare un nuovo metodo nella risorsa `/pets`.

```
aws apigateway put-method --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method POST \
  --authorization-type "NONE" \
```

Questo metodo ti consentirà di inviare informazioni sugli animali domestici alla tabella DynamoDB che hai creato nello stack. AWS CloudFormation

2. Utilizzate il seguente comando per creare un' Servizio AWS integrazione sul metodo. `POST`

```
aws apigateway put-integration --rest-api-id abc123 \
  --resource-id efg456 \
```

```
  --http-method POST \
  --type AWS \
  --integration-http-method POST \
  --uri "arn:aws:apigateway:us-east-2:dynamodb:action/PutItem" \
  --credentials arn:aws:iam::111122223333:role/data-transformation-tutorial-cli-
APIGatewayRole-ABCDEFG \
  --request-templates '{"application/json":"{\"TableName\":\"data-transformation-
tutorial-cli-ddb\",\"Item\":{\"id\":{\"N\":$input.json(\"$.id\")},\"type\":{\"S\":
$input.json(\"$.type\")},\"price\":{\"N\":$input.json(\"$.price\")} }}"}'
```

3. Utilizzare il comando seguente per creare una risposta del metodo per una corretta chiamata del metodo POST.

```
aws apigateway put-method-response --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method POST \
  --status-code 200
```

4. Utilizzare il comando seguente per creare una risposta dell'integrazione per una corretta chiamata del metodo POST.

```
aws apigateway put-integration-response --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method POST \
  --status-code 200 \
  --selection-pattern "" \
  --response-templates '{"application/json": "{\"message\": \"Your response was
 recorded at $context.requestTime\"}"}'
```

Test del metodo **POST**

- Utilizzare il seguente comando per eseguire il test del metodo POST.

```
aws apigateway test-invoke-method --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method POST \
  --body '{\"id\": \"4\", \"type\": \"dog\", \"price\": \"321\"}'
```

L'output mostrerà il messaggio di operazione riuscita.

Per eliminare una AWS CloudFormation pila

- Usa il seguente comando per eliminare le tue AWS CloudFormation risorse.

```
aws cloudformation delete-stack  --stack-name data-transformation-tutorial-cli
```

AWS CloudFormation Modello di trasformazione dei dati completato

L'esempio seguente è un AWS CloudFormation modello completo, che crea una tabella DynamoDB API e una tabella DynamoDB con `/pets` una risorsa `GET` con metodi and. `POST`

- Il `GET` metodo otterrà i dati dall'endpoint. `http://petstore-demo-endpoint.execute-api.com/petstore/pets` HTTP I dati di output verranno trasformati in base al modello di mappatura in PetStore modello di mappatura.
- Il metodo `POST` consentirà all'utente di pubblicare (`POST`) le informazioni sugli animali domestici in una tabella DynamoDB utilizzando un modello di mappatura.

```
AWSTemplateFormatVersion: 2010-09-09
Description: A completed Amazon API Gateway REST API that uses non-proxy integration
 to POST to an Amazon DynamoDB table and non-proxy integration to GET transformed pets
 data.
Parameters:
  StageName:
    Type: String
    Default: v1
    Description: Name of API stage.
Resources:
  DynamoDBTable:
    Type: 'AWS::DynamoDB::Table'
    Properties:
      TableName: !Sub data-transformation-tutorial-complete
      AttributeDefinitions:
        - AttributeName: id
          AttributeType: N
      KeySchema:
        - AttributeName: id
          KeyType: HASH
      ProvisionedThroughput:
        ReadCapacityUnits: 5
        WriteCapacityUnits: 5
```

```
  APIGatewayRole:
    Type: 'AWS::IAM::Role'
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Action:
              - 'sts:AssumeRole'
            Effect: Allow
            Principal:
              Service:
                - apigateway.amazonaws.com
      Policies:
        - PolicyName: APIGatewayDynamoDBPolicy
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                  - 'dynamodb:PutItem'
                Resource: !GetAtt DynamoDBTable.Arn
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Name: data-transformation-complete-api
      ApiKeySourceType: HEADER
  PetsResource:
    Type: 'AWS::ApiGateway::Resource'
    Properties:
      RestApiId: !Ref Api
      ParentId: !GetAtt Api.RootResourceId
      PathPart: 'pets'
  PetsMethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref PetsResource
      HttpMethod: GET
      ApiKeyRequired: false
      AuthorizationType: NONE
      Integration:
        Type: HTTP
        Credentials: !GetAtt APIGatewayRole.Arn
        IntegrationHttpMethod: GET
```

```
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
        PassthroughBehavior: WHEN_NO_TEMPLATES
        IntegrationResponses:
          - StatusCode: '200'
            ResponseTemplates:
              application/json: "#set($inputRoot = $input.path(\"$
\"))\n[\n#foreach($elem in $inputRoot)\n {\n  \"description\": \"Item $elem.id is a
 $elem.type\",\n  \"askingPrice\": \"$elem.price\"\n }#if($foreach.hasNext),#end\n
\n#end\n]"
      MethodResponses:
        - StatusCode: '200'
  PetsMethodPost:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref PetsResource
      HttpMethod: POST
      ApiKeyRequired: false
      AuthorizationType: NONE
      Integration:
        Type: AWS
        Credentials: !GetAtt APIGatewayRole.Arn
        IntegrationHttpMethod: POST
        Uri: arn:aws:apigateway:us-west-1:dynamodb:action/PutItem
        PassthroughBehavior: NEVER
        RequestTemplates:
          application/json: "{\"TableName\":\"data-transformation-tutorial-complete
\",\"Item\":{\"id\":{\"N\":$input.json(\"$.id\")},\"type\":{\"S\":$input.json(\"$.type
\")},\"price\":{\"N\":$input.json(\"$.price\")} }}"
        IntegrationResponses:
          - StatusCode: 200
            ResponseTemplates:
              application/json: "{\"message\": \"Your response was recorded at
 $context.requestTime\"}"
      MethodResponses:
        - StatusCode: '200'

  ApiDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn:
      - PetsMethodGet
    Properties:
      RestApiId: !Ref Api
      StageName: !Sub '${StageName}'
```

```
Outputs:
  ApiId:
    Description: API ID for CLI commands
    Value: !Ref Api
  ResourceId:
    Description: /pets resource ID for CLI commands
    Value: !Ref PetsResource
  ApiRole:
    Description: Role ID to allow API Gateway to put and scan items in DynamoDB table
    Value: !Ref APIGatewayRole
  DDBTableName:
    Description: DynamoDB table name
    Value: !Ref DynamoDBTable
```

Passaggi successivi

Per esplorare modelli di mappatura più complessi, fai riferimento ai seguenti modelli:

- Visualizzare modelli e modelli di mappatura più complessi con l'album fotografico di esempio Esempio di album fotografico.
- Per ulteriori informazioni sui modelli , consulta Modelli di dati per REST APIs.
- Per informazioni su come mappare i diversi output del codice di risposta, Imposta le mappature dei dati di richiesta e risposta utilizzando la console Gateway API.
- Per informazioni su come impostare le mappature dei dati dai dati di richiesta API del metodo an,. APIModello di mappatura del gateway e riferimento alla variabile di registrazione degli accessi

Utilizza un modello di mappatura per sovrascrivere i parametri API di richiesta e risposta e i codici di stato di an

I modelli di mappatura dei parametri e dei codici di risposta Standard di API Gateway consentono di mappare i parametri one-to-one e mappare una famiglia di codici di stato di risposta di integrazione, abbinati a un'espressione regolare, a un unico codice di stato di risposta. Le sostituzioni dei modelli di mappatura consentono di effettuare le seguenti operazioni.

- Esegui mappature dei parametri many-to-one
- Sovrascrivi i parametri dopo l'applicazione delle mappature standard del API Gateway
- Mappa i parametri in modo condizionale in base al contenuto del corpo o ad altri valori dei parametri

- Crea nuovi parametri in modo programmatico
- Sostituisci i codici di stato restituiti dall'endpoint di integrazione

Qualsiasi tipo di parametro di richiesta, intestazione di risposta o codice di stato della risposta può essere sovrascritto.

Casi d'uso per sovrascrivere i parametri API di richiesta e risposta e i codici di stato dell'utente

Nel tuoAPI, potresti utilizzare un modello di mappatura sostitutivo per API eseguire le seguenti operazioni.

- Per creare una nuova intestazione (o sostituire un'intestazione esistente) come una concatenazione di due parametri
- Per sovrascrivere il codice di risposta in a un codice di esito positivo o di errore in base al contenuto del corpo
- Per rimappare con riserva un parametro in base al suo contenuto o al contenuto di alcuni altri parametri
- Per eseguire iterazioni sul contenuto di un corpo json e rimappare coppie chiave-valore a intestazioni o stringhe di query

Considerazioni per sovrascrivere i parametri API di richiesta e risposta e i codici di stato dell'utente

Le seguenti considerazioni potrebbero influire sull'utilizzo delle sostituzioni dei modelli di mappatura.

- Le sovrascritture modello di mappatura non possono essere utilizzate con endpoint di integrazione proxy, che sono privi di mappature dati. Per ulteriori informazioni sui tipi di integrazione, consulta Scegliere un tipo di integrazione API Gateway API.
- Le sovrascritture sono finali. Una sovrascrittura può essere applicata a ciascun parametro una sola volta. Se si tenta di sovrascrivere lo stesso parametro più volte, API Gateway restituisce una risposta. `5XX` Se occorre sovrascrivere lo stesso parametro più volte in tutto il modello, ti consigliamo di creare una variabile e applicare la sovrascrittura alla fine del modello. Il modello viene applicato solo dopo l'analisi dell'intero modello. Per ulteriori informazioni, consulta Sostituisci i parametri e API le intestazioni della richiesta di an utilizzando AWS Management Console

Il modello di mappatura sostituisce il riferimento

Per creare una sovrascrittura modello di mappatura, utilizza una o più delle seguenti variabili`$context` in un modello di mappatura:

Modello di mappatura corpo della richiesta	Modello di mappatura corpo della risposta
`$context.requestOverride.header.` *`header_name`*	`$context.responseOverride.header.` *`header_name`*
`$context.requestOverride.path.` *`path_name`*	`$context.responseOverride.status`
`$context.requestOverride.querystring.` *`querystring_name`*	Nessuno

Per esempi di sovrascrittura dei parametri API di richiesta e risposta e dei codici di stato di an, vedi. the section called “Esempi di sovrascrittura dei parametri API di richiesta e risposta e dei codici di stato di an”

Esempi di sovrascrittura dei parametri API di richiesta e risposta e dei codici di stato di an

Gli esempi seguenti mostrano come creare e testare l'override di un modello di mappatura. I primi due esempi utilizzano AWS Management Console e l'Example API come punto di partenza. Gli ultimi due esempi utilizzano la AWS CLI e la SDK for JavaScript.

Sostituisci il codice API di stato della risposta di an utilizzando il AWS Management Console

Per recuperare un animale domestico utilizzando l' PetStore esempioAPI, si utilizza il API metodo request of`GET /pets/{petId}`, where `{petId}` è un parametro di percorso che può richiedere un numero in fase di esecuzione.

In questo esempio, sovrascriverai il codice di risposta di questo `GET` metodo creando un modello di mappatura mappato `$context.responseOverride.status` a `400` quando viene rilevata una condizione di errore.

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. In APIs, scegli PetStore API, quindi scegli Risorse.
3. Nella struttura Risorse, scegli il metodo `GET` in `/{petId}`.

4. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
5. Per petId, inserisci **`-1`**, quindi scegli Test.

   Nei risultati, puoi notare due cose:

   Innanzitutto, il corpo della risposta indica un out-of-range errore:

```
{
  "errors": [
    {
      "key": "GetPetRequest.petId",
      "message": "The value is out of range."
    }
  ]
}
```

   In secondo luogo, l'ultima riga nella casella Log termina con: `Method completed with status: 200`.
6. Nella scheda Risposta di integrazione scegli Modifica per Predefinito - Risposta.
7. Scegli Modelli di mappatura.
8. Scegliere Add mapping template (Aggiungi modello di mappatura).
9. Per Tipo di contenuto inserisci **`application/json`**.
10. Per Corpo del modello inserisci quanto segue:

```
#set($inputRoot = $input.path('$'))
$input.json("$")
#if($inputRoot.toString().contains("error"))
#set($context.responseOverride.status = 400)
#end
```

11. Selezionare Salva.
12. Seleziona la scheda Test.
13. In petId, immettere **`-1`**.
14. Nei risultati, il Response Body indica un out-of-range errore:

```
{
  "errors": [
```

```
    {
      "key": "GetPetRequest.petId",
      "message": "The value is out of range."
    }
  ]
}
```

Tuttavia, l'ultima riga nella casella Logs (Log) termina ora con: `Method completed with status: 400`.

Sostituisci i parametri e API le intestazioni della richiesta di an utilizzando AWS Management Console

In questo esempio, sovrascriverai il codice dell'intestazione della richiesta del `GET` metodo creando un modello di mappatura mappato `$context.requestOverride.header.`*`header_name`* a una nuova intestazione che combina altre due intestazioni.

1. Accedi alla console Gateway all'indirizzo https://console.aws.amazon.com/apigateway. API
2. In APIs, scegli il PetStore API.
3. Nella struttura Risorse, scegli il metodo `GET` in `/pet`.
4. Nella scheda Richiesta metodo scegli Modifica in Impostazioni della richiesta del metodo.
5. Scegli le intestazioni della HTTP richiesta, quindi scegli Aggiungi intestazione.
6. Per Nome, immetti **`header1`**.
7. Scegli Aggiungi intestazione, quindi crea una seconda intestazione chiamata **`header2`**.
8. Selezionare Salva.
9. Nella scheda Richiesta di integrazione seleziona Modifica per Impostazioni della richiesta di integrazione.
10. Per Richiesta corpo passthrough scegli Quando non ci sono modelli definiti (consigliato).
11. Scegli Modelli di mappatura e procedi come indicato di seguito:

    a. Scegliere Add mapping template (Aggiungi modello di mappatura).

    b. Per Tipo di contenuto inserisci **`application/json`**.

    c. Per Corpo del modello inserisci quanto segue:

```
#set($header1Override = "foo")
#set($header3Value = "$input.params('header1')$input.params('header2')")
$input.json("$")
```

```
#set($context.requestOverride.header.header3 = $header3Value)
#set($context.requestOverride.header.header1 = $header1Override)
#set($context.requestOverride.header.multivalueheader=[$header1Override,
 $header3Value])
```

12. Selezionare Salva.
13. Seleziona la scheda Test.
14. In Headers (Intestazioni) per {pets}, copiare il codice seguente:

```
header1:header1Val
header2:header2Val
```

15. Scegli Test (Esegui test).

    Nel log è presente una voce che include questo testo:

```
Endpoint request headers: {header3=header1Valheader2Val,
header2=header2Val, header1=foo, x-amzn-apigateway-api-id=<api-id>,
Accept=application/json, multivalueheader=foo,header1Valheader2Val}
```

Sostituisci i parametri e le intestazioni API della richiesta di an utilizzando AWS CLI

L'CLIesempio seguente mostra come utilizzare il `put-integration` comando per sovrascrivere un codice di risposta:

```
aws apigateway put-integration --rest-api-id <API_ID> --resource-
id <PATH_TO_RESOURCE_ID> --http-method <METHOD>
            --type <INTEGRATION_TYPE> --request-templates <REQUEST_TEMPLATE_MAP>
```

dove *<REQUEST_TEMPLATE_MAP>* è una mappa dal tipo di contenuto a una stringa del modello da applicare. La struttura della mappa è la seguente:

```
Content_type1=template_string,Content_type2=template_string
```

oppure, nella JSON sintassi:

```
{"content_type1": "template_string"
...}
```

L'esempio seguente mostra come utilizzare il `put-integration-response` comando per sovrascrivere il codice API di risposta di an:

```
aws apigateway put-integration-response --rest-api-id <API_ID> --resource-
id <PATH_TO_RESOURCE_ID> --http-method <METHOD>
            --status-code <STATUS_CODE> --response-templates <RESPONSE_TEMPLATE_MAP>
```

dove *<RESPONSE_TEMPLATE_MAP>* ha lo stesso formato di *<REQUEST_TEMPLATE_MAP>* sopra.

Sovrascrivi i parametri e le intestazioni API della richiesta di an usando il form SDK JavaScript

Nell'esempio seguente viene mostrato come utilizzare il comando `put-integration` per sovrascrivere un codice di risposta:

Richiesta:

```
var params = {
  httpMethod: 'STRING_VALUE', /* required */
  resourceId: 'STRING_VALUE', /* required */
  restApiId: 'STRING_VALUE', /* required */
  type: HTTP | AWS | MOCK | HTTP_PROXY | AWS_PROXY, /* required */
  requestTemplates: {
    '<Content_type>': 'TEMPLATE_STRING',
    /* '<String>': ... */
  },
};
apigateway.putIntegration(params, function(err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else     console.log(data);           // successful response
});
```

Risposta:

```
var params = {
  httpMethod: 'STRING_VALUE', /* required */
  resourceId: 'STRING_VALUE', /* required */
  restApiId: 'STRING_VALUE', /* required */
  statusCode: 'STRING_VALUE', /* required */
  responseTemplates: {
    '<Content_type>': 'TEMPLATE_STRING',
    /* '<String>': ... */
  },
```

```
};
apigateway.putIntegrationResponse(params, function(err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else     console.log(data);           // successful response
});
```

Imposta le mappature dei dati di richiesta e risposta utilizzando la console Gateway API

Per utilizzare la console API Gateway per definire la richiesta/risposta API di integrazione, segui queste istruzioni.

> **Note**
>
> Queste istruzioni presuppongono che sia già stata completata la procedura in Configurazione di una richiesta di integrazione API tramite la console API Gateway.

1. Nel riquadro Risorse, scegli il tuo metodo.
2. Nella scheda Richiesta di integrazione scegli Modifica in Impostazioni della richiesta di integrazione.
3. Scegliete un'opzione per Request body passthrough per configurare in che modo il corpo della richiesta del metodo di un tipo di contenuto non mappato verrà passato attraverso la richiesta di integrazione senza trasformazione nella funzione Lambda, HTTP proxy o proxy di servizio. AWS Sono disponibili tre opzioni:

   - Seleziona Quando nessun modello corrisponde all'intestazione Content-Type della richiesta se vuoi che il corpo della richiesta di metodo venga passato attraverso la richiesta di integrazione al back-end senza trasformazione quando il tipo di contenuto della richiesta di metodo non corrisponde a nessuno dei tipi di contenuto associati ai modelli di mappatura, come illustrato nella prossima fase.

     > **Note**
     >
     > Quando chiamate il API GatewayAPI, scegliete questa opzione impostandola `WHEN_NO_MATCH` come valore della `passthroughBehavior` proprietà nella risorsa di integrazione.

- Selezionare When there are no templates defined (recommended) (Quando non ci sono modelli definiti (consigliato)) se di desidera che il corpo della richiesta di metodo venga passato attraverso la richiesta di integrazione al back-end senza trasformazione quando nella richiesta di integrazione non è stato definito un modello di mappatura. Se viene definito un modello quando viene selezionata questa opzione, la richiesta del metodo di un tipo di contenuto non mappato verrà rifiutata con una risposta HTTP 415 Unsupported Media Type.

> **Note**
>
> Quando chiamate il API GatewayAPI, scegliete questa opzione impostandola `WHEN_NO_TEMPLATE` come valore della `passthroughBehavior` proprietà nella risorsa di integrazione.

- Selezionare Never (Mai) se non si desidera che la richiesta di metodo venga passata quando il tipo di contenuto della richiesta di metodo non corrisponde a nessuno dei tipi di contenuto associati ai modelli di mappatura definiti nella richiesta di integrazione o quando nessun modello di mappatura viene definito nella richiesta di integrazione. La richiesta del metodo di un tipo di contenuto non mappato verrà rifiutata con una risposta HTTP 415 Unsupported Media Type.

> **Note**
>
> Quando chiamate il API GatewayAPI, scegliete questa opzione impostandola `NEVER` come valore della `passthroughBehavior` proprietà nella risorsa di integrazione.

Per ulteriori informazioni sui comportamenti passthrough di integrazione, consulta Comportamenti passthrough di integrazione.

4. Per un HTTP proxy o un proxy di AWS servizio, per associare un parametro path, un parametro della stringa di query o un parametro di intestazione definito nella richiesta di integrazione con un parametro path, un parametro di stringa di query o un parametro di intestazione corrispondente nella richiesta del metodo del HTTP proxy o del proxy di AWS servizio, procedi come segue:

   a. Scegliete rispettivamente i parametri del URL percorso, i parametri della stringa di URL query o le intestazioni della HTTP richiesta, quindi scegliete rispettivamente Aggiungi percorso, Aggiungi stringa di query o Aggiungi intestazione.

b. Per Nome, digitate il nome del parametro path, del parametro della stringa di query o del parametro di intestazione nel proxy o AWS nel HTTP proxy del servizio.

c. Per Mappato da immetti il valore della mappatura del parametro di percorso, del parametro di stringa di query o del parametro di intestazione. Utilizza uno dei seguenti formati:

- **`method.request.path.`*`parameter-name`***per un parametro di percorso denominato *`parameter-name`* come definito nella pagina di richiesta del metodo.
- **`method.request.querystring.`*`parameter-name`***per un parametro della stringa di query denominato *`parameter-name`* come definito nella pagina di richiesta del metodo.
- **`method.request.multivaluequerystring.`*`parameter-name`***per un parametro di stringa di query multivalore denominato *`parameter-name`* come definito nella pagina di richiesta del metodo.
- **`method.request.header.`*`parameter-name`***per un parametro di intestazione denominato *`parameter-name`* come definito nella pagina di richiesta del metodo.

  In alternativa, puoi impostare un valore letterale della stringa (inserito in una coppia di virgolette singole) su un'intestazione di integrazione.
- **`method.request.multivalueheader.`*`parameter-name`***per un parametro di intestazione multivalore denominato *`parameter-name`* come definito nella pagina di richiesta del metodo.

d. Per aggiungere un altro parametro, scegli il pulsante Aggiungi.

5. Per aggiungere un modello di mappatura, scegli Modelli di mappatura.
6. Per definire un modello di mappatura per una richiesta in ingresso, scegli Aggiungi modello di mappatura. Per Tipo di contenuto immetti un tipo di contenuto (ad esempio, **`application/json`**). Quindi, inserisci il modello di mappatura. Per ulteriori informazioni, consulta Modelli di mappatura per REST APIs.
7. Seleziona Save (Salva.
8. È possibile mappare una risposta di integrazione dal backend a una risposta di metodo dell'app API restituita all'app chiamante. Ciò include la restituzione al client delle intestazioni di risposta selezionate tra quelle disponibili dal backend, la trasformazione del formato dei dati del payload di risposta del backend in un formato specificato. API Puoi specificare la mappatura configurando Risposta metodo e Risposte di integrazione.

Per fare in modo che il metodo riceva un formato di dati di risposta personalizzato basato sul codice di HTTP stato restituito dalla funzione Lambda, dal proxy o dal HTTP proxy di AWS servizio, procedi come segue:

a. Scegli Risposte di integrazione. Scegliete Modifica sul valore predefinito - Risposta per specificare le impostazioni per un codice di HTTP risposta di 200 dal metodo, oppure scegliete Crea risposta per specificare le impostazioni per qualsiasi altro codice di stato della HTTP risposta del metodo.

b. Per l'espressione regolare di errore Lambda (per una funzione Lambda) o l'espressione regolare di HTTP stato (per un HTTP proxy o un proxy di AWS servizio), immettete un'espressione regolare per specificare quali stringhe di errore della funzione Lambda (per una funzione Lambda) o codici di stato della HTTP risposta (per un proxy o proxy di servizio) vengono mappate a questa mappatura di output. HTTP AWS Ad esempio, per mappare tutti i codici di stato della HTTP risposta 2xx da un HTTP proxy a questa mappatura di output, digitate "" come espressione regolare di stato. **`2\d{2}`** HTTP Per restituire un messaggio di errore contenente "Invalid Request" da una funzione Lambda a una risposta `400 Bad Request`, digita "**`.*Invalid request.*`**" per Espressione regolare errore Lambda. Per restituire invece `400 Bad Request` per tutti i messaggi di errore non mappati da Lambda, immetti "**`(\n|.)+`**" in Espressione regolare errore Lambda. Quest'ultima espressione regolare può essere utilizzata per la risposta di errore predefinita di un. API

> **Note**
>
> APIGateway utilizza espressioni regolari in stile pattern Java per la mappatura delle risposte. Per ulteriori informazioni, consulta Pattern nella documentazione Oracle.
> I modelli di errore vengono confrontati con la stringa intera della proprietà `errorMessage` nella risposta Lambda, popolata da `callback(errorMessage)` in Node.js o da `throw new MyException(errorMessage)` in Java. Inoltre, i caratteri con escape sono senza escape prima che venga applicata l'espressione regolare.
> Se si utilizza "+" come modello di selezione per filtrare le risposte, considera che potrebbe non corrispondere a una risposta contenente un carattere di nuova riga ("\n").

c. Se abilitato, per Method response status, selezionate il codice di stato della HTTP risposta definito nella pagina Method response.

d. Per le mappature degli header, per ogni intestazione definita per il codice di stato della HTTP risposta nella pagina Method response, specificate un valore di mappatura. Per Mapping valore (Valore mappatura), utilizzare uno dei seguenti formati:

   - **`integration.response.multivalueheaders.`*`header-name`***dove *`header-name`* è il nome di un'intestazione di risposta multivalore del backend.

     Ad esempio, per restituire l'intestazione della risposta del backend come **`Date`** intestazione della risposta di un API metodo, la colonna di intestazione della risposta conterrà una voce Timestamp e il valore di mappatura associato deve essere impostato su Integration.Response.MultivalueHeaders.date. **`Timestamp`**

   - **`integration.response.header.`*`header-name`***dove *`header-name`* è il nome di un'intestazione di risposta a valore singolo proveniente dal backend.

     Ad esempio, per restituire l'intestazione della risposta del backend come **`Date`** intestazione della risposta di un API metodo, la colonna di intestazione della **`Timestamp`** risposta conterrà una voce Timestamp e il valore di mappatura associato deve essere impostato su Integration.Response.Header.Date.

e. Scegli Modelli di mappatura, quindi seleziona Aggiungi modello di mappatura. Nella casella Tipo di contenuto, inserisci il tipo di contenuto dei dati che verranno passati dalla funzione Lambda, dal proxy o dal HTTP proxy di AWS servizio al metodo. Quindi, inserisci il modello di mappatura. Per ulteriori informazioni, consulta Modelli di mappatura per REST APIs.

f. Seleziona Save (Salva.

Esempi di modelli di dati e modelli di mappatura per Gateway API

Le sezioni seguenti forniscono esempi di modelli e modelli di mappatura che possono essere utilizzati come punto di partenza per creare progetti personalizzati APIs in API Gateway. Per ulteriori informazioni sulle trasformazioni dei dati, consultare Modelli di mappatura per REST APIs. Per ulteriori informazioni sui modelli di dati, vedereThe section called “Modelli di dati per REST APIs”.

Argomenti

- Esempio di album fotografico
- Esempio di articolo di notizie

Esempio di album fotografico

L'esempio seguente mostra un album di foto API in API Gateway. Viene fornito un esempio di trasformazione dei dati, modelli aggiuntivi e modelli di mappatura.

Argomenti

- Esempio di trasformazione dei dati
- Modello di input per i dati relativi alle foto
- Modello di output per i dati relativi alle foto
- Modello di mappatura di input per i dati relativi alle foto

Esempio di trasformazione dei dati

L'esempio seguente mostra come trasformare i dati di input relativi alle foto utilizzando un modello di mappatura Velocity Template Language (VTL). Per ulteriori informazioni sul Velocity Template Language, vedete Apache Velocity - Reference. VTL

L'esempio seguente mostra i dati di input per una richiesta di integrazione.

```
{
  "photos": {
    "page": 1,
    "pages": "1234",
    "perpage": 100,
    "total": "123398",
    "photo": [
      {
        "id": "12345678901",
        "owner": "23456789@A12",
        "photographer_first_name" : "Saanvi",
        "photographer_last_name" : "Sarkar",
        "secret": "abc123d456",
        "server": "1234",
        "farm": 1,
        "title": "Sample photo 1",
        "ispublic": true,
        "isfriend": false,
        "isfamily": false
      },
      {
        "id": "23456789012",
```

```
        "owner": "34567890@B23",
        "photographer_first_name" : "Richard",
        "photographer_last_name" : "Roe",
        "secret": "bcd234e567",
        "server": "2345",
        "farm": 2,
        "title": "Sample photo 2",
        "ispublic": true,
        "isfriend": false,
        "isfamily": false
      }
    ]
  }
}
```

L'esempio seguente è un modello di mappatura per trasformare i dati delle foto.

```
#set($inputRoot = $input.path('$'))
{
  "photos": [
#foreach($elem in $inputRoot.photos.photo)
    {
      "id": "$elem.id",
      "photographedBy": "$elem.photographer_first_name $elem.photographer_last_name",
      "title": "$elem.title",
      "ispublic": $elem.ispublic,
      "isfriend": $elem.isfriend,
      "isfamily": $elem.isfamily
    }#if($foreach.hasNext),#end

#end
  ]
}
```

L'esempio seguente mostra i dati di output della trasformazione.

```
{
  "photos": [
    {
      "id": "12345678901",
      "photographedBy": "Saanvi Sarkar",
      "title": "Sample photo 1",
      "ispublic": true,
```

```
      "isfriend": false,
      "isfamily": false
    },
    {
      "id": "23456789012",
      "photographedBy": "Richard Roe",
      "title": "Sample photo 2",
      "ispublic": true,
      "isfriend": false,
      "isfamily": false
    }
  ]
}
```

Modello di input per i dati relativi alle foto

È possibile definire un modello per i dati di input. Questo modello di input richiede il caricamento di una foto e specifica un minimo di 10 foto per ogni pagina. Puoi utilizzare questo modello di input per generare SDK o attivare una convalida della richiesta per il tuo. API

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "PhotosInputModel",
  "type": "object",
  "properties": {
    "photos": {
      "type": "object",
      "required" : [
      "photo"
      ],
      "properties": {
        "page": { "type": "integer" },
        "pages": { "type": "string" },
        "perpage": { "type": "integer", "minimum" : 10 },
        "total": { "type": "string" },
        "photo": {
          "type": "array",
          "items": {
            "type": "object",
            "properties": {
              "id": { "type": "string" },
              "owner": { "type": "string" },
              "photographer_first_name" : {"type" : "string"},
```

```
                "photographer_last_name" : {"type" : "string"},
                "secret": { "type": "string" },
                "server": { "type": "string" },
                "farm": { "type": "integer" },
                "title": { "type": "string" },
                "ispublic": { "type": "boolean" },
                "isfriend": { "type": "boolean" },
                "isfamily": { "type": "boolean" }
              }
            }
          }
        }
      }
    }
```

Modello di output per i dati relativi alle foto

È possibile definire un modello per i dati di output. È possibile utilizzare questo modello per un modello di risposta al metodo, necessario quando si genera un codice fortemente tipizzato SDK per. API Ciò garantisce che l'output venga trasmesso in una classe appropriata in Java o Objective-C.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "PhotosOutputModel",
  "type": "object",
  "properties": {
    "photos": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "id": { "type": "string" },
          "photographedBy": { "type": "string" },
          "title": { "type": "string" },
          "ispublic": { "type": "boolean" },
          "isfriend": { "type": "boolean" },
          "isfamily": { "type": "boolean" }
        }
      }
    }
  }
}
```

Modello di mappatura di input per i dati relativi alle foto

È possibile definire un modello di mappatura per modificare i dati di input. È possibile modificare i dati di input per ulteriori integrazioni di funzioni o risposte di integrazione.

```
#set($inputRoot = $input.path('$'))
{
  "photos": {
    "page": $inputRoot.photos.page,
    "pages": "$inputRoot.photos.pages",
    "perpage": $inputRoot.photos.perpage,
    "total": "$inputRoot.photos.total",
    "photo": [
#foreach($elem in $inputRoot.photos.photo)
      {
        "id": "$elem.id",
        "owner": "$elem.owner",
        "photographer_first_name" : "$elem.photographer_first_name",
        "photographer_last_name" : "$elem.photographer_last_name",
        "secret": "$elem.secret",
        "server": "$elem.server",
        "farm": $elem.farm,
        "title": "$elem.title",
        "ispublic": $elem.ispublic,
        "isfriend": $elem.isfriend,
        "isfamily": $elem.isfamily
      }#if($foreach.hasNext),#end

#end
    ]
  }
}
```

Esempio di articolo di notizie

L'esempio seguente mostra un articolo di notizie API in API Gateway. Viene fornito un esempio di trasformazione dei dati, modelli aggiuntivi e modelli di mappatura.

Argomenti

- Esempio di trasformazione dei dati
- Modello di input per i dati delle notizie
- Modello di output per i dati delle notizie

- Modello di mappatura degli input per i dati relativi alle notizie

Esempio di trasformazione dei dati

L'esempio seguente mostra come trasformare i dati di input relativi a un articolo di notizie utilizzando un modello di mappatura Velocity Template Language (VTL). Per ulteriori informazioni sul Velocity Template Language, vedete Apache Velocity - Reference. VTL

L'esempio seguente mostra i dati di input per una richiesta di integrazione.

```
{
  "count": 1,
  "items": [
    {
      "last_updated_date": "2015-04-24",
      "expire_date": "2016-04-25",
      "author_first_name": "John",
      "description": "Sample Description",
      "creation_date": "2015-04-20",
      "title": "Sample Title",
      "allow_comment": true,
      "author": {
        "last_name": "Doe",
        "email": "johndoe@example.com",
        "first_name": "John"
      },
      "body": "Sample Body",
      "publish_date": "2015-04-25",
      "version": "1",
      "author_last_name": "Doe",
      "parent_id": 2345678901,
      "article_url": "http://www.example.com/articles/3456789012"
    }
  ],
  "version": 1
}
```

L'esempio seguente è un modello di mappatura per trasformare i dati delle notizie.

```
#set($inputRoot = $input.path('$'))
{
  "count": $inputRoot.count,
```

```
  "items": [
#foreach($elem in $inputRoot.items)
    {
      "creation_date": "$elem.creation_date",
      "title": "$elem.title",
      "author": "$elem.author.first_name $elem.author.last_name",
      "body": "$elem.body",
      "publish_date": "$elem.publish_date",
      "article_url": "$elem.article_url"
    }#if($foreach.hasNext),#end

#end
  ],
  "version": $inputRoot.version
}
```

L'esempio seguente mostra i dati di output della trasformazione.

```
{
  "count": 1,
  "items": [
    {
      "creation_date": "2015-04-20",
      "title": "Sample Title",
      "author": "John Doe",
      "body": "Sample Body",
      "publish_date": "2015-04-25",
      "article_url": "http://www.example.com/articles/3456789012"
    }
  ],
  "version": 1
}
```

Modello di input per i dati delle notizie

È possibile definire un modello per i dati di input. Questo modello di input richiede che un articolo di notizie contenga unURL, un titolo e un corpo. Puoi utilizzare questo modello di input per generare SDK o attivare una convalida della richiesta per il tuoAPI.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "NewsArticleInputModel",
  "type": "object",
```

```
  "properties": {
    "count": { "type": "integer" },
    "items": {
      "type": "array",
      "items": {
        "type": "object",
        "required": [
      "article_url",
      "title",
      "body"
      ],
        "properties": {
          "last_updated_date": { "type": "string" },
          "expire_date": { "type": "string" },
          "author_first_name": { "type": "string" },
          "description": { "type": "string" },
          "creation_date": { "type": "string" },
          "title": { "type": "string" },
          "allow_comment": { "type": "boolean" },
          "author": {
            "type": "object",
            "properties": {
              "last_name": { "type": "string" },
              "email": { "type": "string" },
              "first_name": { "type": "string" }
            }
          },
          "body": { "type": "string" },
          "publish_date": { "type": "string" },
          "version": { "type": "string" },
          "author_last_name": { "type": "string" },
          "parent_id": { "type": "integer" },
          "article_url": { "type": "string" }
        }
      }
    },
    "version": { "type": "integer" }
  }
}
```

Modello di output per i dati delle notizie

È possibile definire un modello per i dati di output. È possibile utilizzare questo modello per un modello di risposta al metodo, necessario quando si genera un codice fortemente tipizzato SDK per. API Ciò garantisce che l'output venga trasmesso in una classe appropriata in Java o Objective-C.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "PhotosOutputModel",
  "type": "object",
  "properties": {
    "photos": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "id": { "type": "string" },
          "photographedBy": { "type": "string" },
          "title": { "type": "string" },
          "ispublic": { "type": "boolean" },
          "isfriend": { "type": "boolean" },
          "isfamily": { "type": "boolean" }
        }
      }
    }
  }
}
```

Modello di mappatura degli input per i dati relativi alle notizie

È possibile definire un modello di mappatura per modificare i dati di input. È possibile modificare i dati di input per ulteriori integrazioni di funzioni o risposte di integrazione.

```
#set($inputRoot = $input.path('$'))
{
  "count": $inputRoot.count,
  "items": [
#foreach($elem in $inputRoot.items)
    {
      "last_updated_date": "$elem.last_updated_date",
      "expire_date": "$elem.expire_date",
      "author_first_name": "$elem.author_first_name",
```

```
      "description": "$elem.description",
      "creation_date": "$elem.creation_date",
      "title": "$elem.title",
      "allow_comment": "$elem.allow_comment",
      "author": {
        "last_name": "$elem.author.last_name",
        "email": "$elem.author.email",
        "first_name": "$elem.author.first_name"
      },
      "body": "$elem.body",
      "publish_date": "$elem.publish_date",
      "version": "$elem.version",
      "author_last_name": "$elem.author_last_name",
      "parent_id": $elem.parent_id,
      "article_url": "$elem.article_url"
    }#if($foreach.hasNext),#end

#end
  ],
  "version": $inputRoot.version
}
```

Riferimento alla mappatura dei dati di API richiesta e risposta di Amazon API Gateway

Questa sezione spiega come impostare le mappature dei dati dai dati API di una richiesta di metodo, inclusi altri dati archiviati in `context`, o `util`variabili `stage`, ai parametri di richiesta di integrazione corrispondenti e dai dati di risposta di integrazione, inclusi gli altri dati, ai parametri di risposta del metodo. I dati della richiesta del metodo includono i parametri di richiesta (percorso, stringa di query, intestazioni) e il corpo. I dati della risposta di integrazione includono i parametri di risposta (intestazioni) e il corpo. Per ulteriori informazioni sull'uso delle variabili di fase, consulta APIRiferimento alle variabili di fase Gateway per REST APIs in Gateway API.

Argomenti

- Come mappare i dati di richiesta di metodi ai parametri di richiesta di integrazione
- Mappa i dati delle risposte di integrazione alle intestazioni delle risposte di metodo
- Mappa i payload di risposta e di richiesta tra metodo e integrazione
- Comportamenti passthrough di integrazione

Come mappare i dati di richiesta di metodi ai parametri di richiesta di integrazione

I parametri di richiesta di integrazione, sotto forma di variabili di percorso, stringhe o intestazioni di query, possono essere mappati dal payload e da qualsiasi parametro di richiesta di metodo definito.

Nella tabella seguente, *PARAM_NAME* è il nome di un parametro di richiesta del metodo del tipo di parametro specificato. Deve corrispondere all'espressione regolare `'^[a-zA-Z0-9._$-]+$]'`. Deve essere stato definito prima che possa essere referenziato. *JSONPath_EXPRESSION* è un'JSONPathespressione per un JSON campo del corpo di una richiesta o di una risposta.

> **Note**
>
> Il prefisso `"$"` viene omesso in questa sintassi.

Fonte di dati mappata	Espressione di mappatura
Percorso della richiesta di metodo	`method.request.path.` *PARAM_NAME*
Stringa di query della richiesta di metodo	`method.request.querystring.` *PARAM_NAME*
Stringa di query multi-valore della richiesta del metodo	`method.request.multivaluequerystring.` *PARAM_NAME*
Intestazione della richiesta di metodo	`method.request.header.` *PARAM_NAME*
Intestazione multi-valore della richiesta di metodo	`method.request.multivalueheader.` *PARAM_NAME*
Corpo della richiesta di metodo	`method.request.body`
Corpo della richiesta del metodo () JsonPath	`method.request.body.` *JSONPath_EXPRESSION* .
Variabili di fase	`stageVariables.` *VARIABLE_NAME*
Variabili di contesto	`context.`*VARIABLE_NAME* che deve essere una delle variabili di contesto supportate.

Fonte di dati mappata	Espressione di mappatura
Valore statico	`'STATIC_VALUE'` . La `STATIC_VALUE` è una stringa letterale e deve essere racchiusa tra virgolette singole.

Example Mappature dal parametro di richiesta del metodo in Open API

L'esempio seguente mostra uno API snippet Open che mappa:

- l'intestazione della richiesta del metodo, `methodRequestHeaderParam`, al parametro di percorso `integrationPathParam` della richiesta di integrazione
- la stringa di query multi-valore della richiesta del metodo, `methodRequestQueryParam`, alla stringa di query della richiesta di integrazione, `integrationQueryParam`

```
...
"requestParameters" : {

    "integration.request.path.integrationPathParam" :
 "method.request.header.methodRequestHeaderParam",
    "integration.request.querystring.integrationQueryParam" :
 "method.request.multivaluequerystring.methodRequestQueryParam"

}
...
```

I parametri della richiesta di integrazione possono anche essere mappati dai campi del corpo della JSON richiesta utilizzando un'espressione. JSONPath La tabella seguente mostra le espressioni di mappatura per un corpo di richiesta del metodo e i relativi JSON campi.

Example Mappatura dal corpo della richiesta del metodo in Open API

L'esempio seguente mostra uno API snippet Open che mappa 1) il corpo della richiesta del metodo all'intestazione della richiesta di integrazione, denominatabody-header, e 2) un JSON campo del corpo, come espresso da un'JSONespressione (`petstore.pets[0].name`senza il prefisso). `$.`

```
...
"requestParameters" : {

    "integration.request.header.body-header" : "method.request.body",
    "integration.request.path.pet-name" : "method.request.body.petstore.pets[0].name",

}
...
```

Mappa i dati delle risposte di integrazione alle intestazioni delle risposte di metodo

I parametri delle intestazioni delle risposte di metodo possono essere mappati da qualsiasi intestazione o corpo della risposta di integrazione, variabili `$context` o valori statici. La tabella seguente descrive le espressioni di mappatura dell'intestazione della risposta del metodo.

Origine dati mappata	Espressione di mappatura
Intestazione della risposta di integrazione	`integration.response.header.` *`PARAM_NAME`*
Intestazione della risposta di integrazione	`integration.response.multivalueheader.` *`PARAM_NAME`*
Corpo della risposta di integrazione	`integration.response.body`
Corpo della risposta di integrazione () JsonPath	`integration.response.body.` *`JSONPath_EXPRESSION`*
Variabile di fase	`stageVariables.` *`VARIABLE_NAME`*
Variabile di contesto	`context.`*`VARIABLE_NAME` che deve essere una delle variabili di contesto supportate.*
Valore statico	*`'STATIC_VALUE'`* . La *`STATIC_VALUE`* è una stringa letterale e deve essere racchiusa tra virgolette singole.

Example Mappatura dei dati dalla risposta di integrazione in Open API

L'esempio seguente mostra uno API snippet Open che mappa 1) il JSONPath campo della risposta di integrazione`redirect.url`, nell'intestazione della risposta alla richiesta e 2) l'`location`intestazione della risposta di integrazione all'`x-app-id`intestazione della risposta del metodo. `id`

```
...
"responseParameters" : {

    "method.response.header.location" : "integration.response.body.redirect.url",
    "method.response.header.id" : "integration.response.header.x-app-id",
    "method.response.header.items" : "integration.response.multivalueheader.item",

}
...
```

Mappa i payload di risposta e di richiesta tra metodo e integrazione

APIGateway utilizza il motore Velocity Template Language (VTL) per elaborare i modelli di body mapping per la richiesta di integrazione e la risposta di integrazione. I modelli di mappatura traducono i payload delle richieste di metodo nei payload della richiesta di integrazione corrispondenti e i corpi delle risposte di integrazione nei corpi delle risposte di metodo corrispondenti.

I VTL modelli utilizzano JSONPath espressioni, altri parametri come contesti di chiamata e variabili di fase e funzioni di utilità per elaborare i dati. JSON

Se viene definito un modello per descrivere la struttura dei dati di un payload, API Gateway può utilizzare il modello per generare un modello di mappatura scheletrica per una richiesta di integrazione o una risposta di integrazione. È possibile utilizzare il modello scheletrico come ausilio per personalizzare ed espandere lo script di mappatura. VTL Tuttavia, puoi creare un nuovo modello di mappatura senza definire un modello per la struttura dati del payload.

Seleziona un modello di mappatura VTL

APIGateway utilizza la seguente logica per selezionare un modello di mappatura, in Velocity Template Language (VTL), per mappare il payload da una richiesta di metodo alla richiesta di

integrazione corrispondente o per mappare il payload da una risposta di integrazione alla risposta del metodo corrispondente.

Per un payload di richiesta, API Gateway utilizza il valore di `Content-Type` intestazione della richiesta come chiave per selezionare il modello di mappatura per il payload della richiesta. Per un payload di risposta, API Gateway utilizza il valore di `Accept` intestazione della richiesta in entrata come chiave per selezionare il modello di mappatura.

Quando l'`Content-Type`intestazione è assente nella richiesta, API Gateway presume che il suo valore predefinito sia. `application/json` Per tale richiesta, API Gateway utilizza `application/json` come chiave predefinita per selezionare il modello di mappatura, se definito. Quando nessun modello corrisponde a questa chiave, API Gateway passa il payload a unmapped se la passthroughBehaviorproprietà è impostata su o. `WHEN_NO_MATCH` `WHEN_NO_TEMPLATES`

Quando l'`Accept`intestazione non è specificata nella richiesta, API Gateway presume che il suo valore predefinito sia. `application/json` In questo caso, API Gateway seleziona un modello di mappatura esistente per mappare il payload `application/json` di risposta. Se non è definito alcun modello per`application/json`, API Gateway seleziona il primo modello esistente e lo utilizza come predefinito per mappare il payload di risposta. Analogamente, API Gateway utilizza il primo modello esistente quando il valore di `Accept` intestazione specificato non corrisponde a nessuna chiave del modello esistente. Se non viene definito alcun modello, API Gateway passa semplicemente il payload di risposta tramite unmapped.

Ad esempio, supponiamo che un API abbia un `application/json` modello definito per un payload di richiesta e abbia un `application/xml` modello definito per il payload di risposta. Se il client imposta le intestazioni `"Content-Type : application/json"` e `"Accept : application/xml"` nella richiesta, i payload di richiesta e di risposta verranno elaborati con i modelli di mappatura corrispondenti. Se l'intestazione `Accept:application/xml` non è presente, verrà utilizzato il modello di mappatura `application/xml` per mappare il payload della risposta. Al contrario, per restituire il payload di risposta non mappato, devi configurare un modello vuoto per `application/json`.

Quando si seleziona un modello di mappatura, viene utilizzato solo il MIME tipo `Content-Type` contenuto nelle intestazioni `Accept` and. Ad esempio, un'intestazione di `"Content-Type: application/json; charset=UTF-8"` avrà un modello di richiesta con la chiave `application/json` selezionata.

Comportamenti passthrough di integrazione

Con le integrazioni non proxy, quando una richiesta di metodo porta un payload e l'intestazione `Content-Type` non corrisponde a nessun modello di mappatura specificato o non è stato definito un modello di mappatura, puoi decidere di passare il payload della richiesta fornito dal client attraverso la richiesta di integrazione al back-end senza trasformazione. Il processo è noto come passthrough di integrazione.

Per le integrazioni proxy, API Gateway trasmette l'intera richiesta al backend e non è possibile modificare i comportamenti del passthrough.

Il comportamento di passthrough utilizzato per una richiesta in entrata è definito in base all'opzione scelta per un modello di mappatura specifico, durante la configurazione della richiesta di integrazione e dall'intestazione del Tipo di contenuto che un client ha impostato nella richiesta in entrata. Sono disponibili tre opzioni:

Opzione Quando nessun modello corrisponde all'intestazione Content-Type della richiesta

Selezionare questa opzione se si desidera che venga eseguito il passthrough del corpo della richiesta di metodo nella richiesta di integrazione al back-end senza trasformazione quando il tipo di contenuto della richiesta di metodo non corrisponde ai tipi di contenuto associati ai modelli di mappatura.

Quando si chiama il API GatewayAPI, si sceglie questa opzione impostandola `WHEN_NO_MATCH` come valore della `passthroughBehavior` proprietà sull'integrazione.

Quando non ci sono modelli definiti (consigliato)

Selezionare questa opzione se si desidera che venga eseguito il passthrough del corpo della richiesta di metodo nella richiesta di integrazione al back-end senza trasformazione quando nella richiesta di integrazione non è stato definito un modello di mappatura. Se viene definito un modello quando viene selezionata questa opzione, la richiesta del metodo di un tipo di contenuto non mappato verrà rifiutata con una risposta HTTP 415 Unsupported Media Type.

Quando si chiama il API GatewayAPI, si sceglie questa opzione impostandola `WHEN_NO_TEMPLATES` come valore della `passthroughBehavior` proprietà nell'Integrazione.

Mai

Selezionare questa opzione se non si desidera che venga eseguito il passthrough del corpo della richiesta di metodo nella richiesta di integrazione al back-end senza trasformazione quando

nella richiesta di integrazione non è stato definito un modello di mappatura. Se viene definito un modello quando viene selezionata questa opzione, la richiesta del metodo di un tipo di contenuto non mappato verrà rifiutata con una risposta HTTP 415 Unsupported Media Type.

Quando si chiama il API GatewayAPI, si sceglie questa opzione impostandola `NEVER` come valore della `passthroughBehavior` proprietà nell'Integrazione.

Negli esempi seguenti vengono illustrati i possibili comportamenti di passthrough.

Esempio 1: viene definito un modello di mappatura nella richiesta di integrazione per il tipo di contenuto `application/json`.

Intestazione tipo contenuto\Opzione passthrough selezionata	**`WHEN_NO_MATCH`**	**`WHEN_NO_TEMPLATES`**	**`NEVER`**
Nessuna (impostazione predefinita su `application/json`)	Il payload della richiesta viene trasformato utilizzando il modello.	Il payload della richiesta viene trasformato utilizzando il modello.	Il payload della richiesta viene trasformato utilizzando il modello.
`application/json`	Il payload della richiesta viene trasformato utilizzando il modello.	Il payload della richiesta viene trasformato utilizzando il modello.	Il payload della richiesta viene trasformato utilizzando il modello.
`application/xml`	Il payload della richiesta non viene trasformato e viene inviato al back-end inalterato.	La richiesta viene rifiutata con una HTTP `415 Unsupported Media Type` risposta.	La richiesta viene rifiutata con una HTTP `415 Unsupported Media Type` risposta.

Esempio 2: viene definito un modello di mappatura nella richiesta di integrazione per il tipo di contenuto `application/xml`.

Intestazione tipo contenuto\Opzione passthrough selezionata	**WHEN_NO_MATCH**	**WHEN_NO_TEMPLATES**	**NEVER**
Nessuna (impostazione predefinita su `application/json`)	Il payload della richiesta non viene trasformato e viene inviato al back-end inalterato.	La richiesta viene rifiutata con una HTTP `415 Unsupported Media Type` risposta.	La richiesta viene rifiutata con una HTTP `415 Unsupported Media Type` risposta.
`application/json`	Il payload della richiesta non viene trasformato e viene inviato al back-end inalterato.	La richiesta viene rifiutata con una HTTP `415 Unsupported Media Type` risposta.	La richiesta viene rifiutata con una HTTP `415 Unsupported Media Type` risposta.
`application/xml`	Il payload della richiesta viene trasformato utilizzando il modello.	Il payload della richiesta viene trasformato utilizzando il modello.	Il payload della richiesta viene trasformato utilizzando il modello.

APIModello di mappatura del gateway e riferimento alla variabile di registrazione degli accessi

Questa sezione fornisce informazioni di riferimento per le variabili e le funzioni che Amazon API Gateway definisce per l'uso con modelli di dati, autorizzatori, modelli di mappatura e registrazione degli CloudWatch accessi. Per informazioni dettagliate su come usare queste variabili e funzioni, consulta Modelli di mappatura per REST APIs. Per ulteriori informazioni sul Velocity Template Language (VTL), consulta la pagina di riferimento. VTL

Argomenti

- $contextVariabili per modelli di dati, autorizzatori, modelli di mappatura e registrazione degli accessi CloudWatch

- Esempio di modello di variabile $context
- Variabili $context solo per la registrazione degli accessi
- $input Variabili
- Esempi di modello di variabile $input
- $stageVariables
- $util Variabili

> **Note**
>
> Per le variabili `$method` e `$integration`, consulta the section called “Riferimento alla mappatura dei dati di richiesta e di risposta”.

`$context`Variabili per modelli di dati, autorizzatori, modelli di mappatura e registrazione degli accessi CloudWatch

Le seguenti `$context` variabili possono essere utilizzate nei modelli di dati, negli autorizzatori, nei modelli di mappatura e nella registrazione degli accessi. CloudWatch APIGateway potrebbe aggiungere variabili di contesto aggiuntive.

Per `$context` le variabili che possono essere utilizzate solo nella registrazione CloudWatch degli accessi, vedereThe section called “Variabili `$context` solo per la registrazione degli accessi”.

Parametro	Descrizione
`$context.accountId`	L'ID dell' AWS account del API proprietario.
`$context.apiId`	L'identificatore che API Gateway assegna al tuo. API
`$context.authorizer.claims.` *`property`*	Proprietà delle richieste restituite dal pool di utenti di Amazon Cognito dopo che l'interme diario del metodo viene autenticato correttam ente. Per ulteriori informazioni, consulta the section called “Utilizza un pool di utenti di Amazon Cognito come autorizzazione per un'API REST”.

Parametro	Descrizione
	Note La chiamata di `$context.authorizer.claims` restituisce null.
`$context.authorizer.principalId`	L'identificazione dell'utente principale associata al token inviato dal client e restituito da un autorizzatore API Gateway Lambda (precedentemente noto come autorizzatore personalizzato). Per ulteriori informazioni, consulta the section called "Uso di autorizzazioni Lambda".
`$context.authorizer.`*`property`*	Il valore stringato della coppia chiave-valore specificata della `context` mappa restituito da una funzione di autorizzazione Gateway API Lambda. Ad esempio, se le autorizzazioni restituiscono la mappa `context` seguente: `"context" : { "key": "value", "numKey": 1, "boolKey": true }` la chiamata di `$context.authorizer.key` restituisce la stringa `"value"`, la chiamata di `$context.authorizer.numKey` restituisce la stringa `"1"` e la chiamata di `$context.authorizer.boolKey` restituisce la stringa `"true"`. Per ulteriori informazioni, consulta the section called "Uso di autorizzazioni Lambda".
`$context.awsEndpointRequestId`	AWS L'ID della richiesta dell'endpoint.

Parametro	Descrizione
`$context.deploymentId`	L'ID della API distribuzione.
`$context.domainName`	Il nome di dominio completo utilizzato per richiamare ilAPI. Deve essere lo stesso dell'inte stazione `Host` in ingresso.
`$context.domainPrefix`	Prima etichetta di `$context.domainName` .
`$context.error.message`	Una stringa contenente un messaggio di errore API Gateway. Questa variabile può essere utilizzata solo per la semplice sostituzi one di variabili in un modello di GatewayRe sponsebody mapping, che non viene elaborato dal motore Velocity Template Language, e nella registrazione degli accessi. Per ulteriori informazioni, consulta the section called “Metriche” e the section called “Configurazione delle risposte del gateway per la personali zzazione delle risposte agli errori”.
`$context.error.messageString`	Valore `$context.error.message` tra virgolette, ovvero `"$context.error.me ssage"` .
`$context.error.responseType`	Un tipo di. GatewayResponse Questa variabile può essere utilizzata solo per la semplice sostituzione di variabili in un modello di GatewayResponsebody mapping, che non viene elaborato dal motore Velocity Template Language, e nella registrazione degli accessi. Per ulteriori informazioni, consulta the section called “Metriche” e the section called “Configur azione delle risposte del gateway per la personalizzazione delle risposte agli errori”.

Parametro	Descrizione
`$context.error.validationErrorString`	Stringa contenente un messaggio dettagliato di errore di convalida.
`$context.extendedRequestId`	L'ID esteso che API Gateway genera e assegna alla richiesta. API L'ID della richiesta esteso contiene ulteriori informazioni utili per il debug e la risoluzione dei problemi.
`$context.httpMethod`	Il HTTP metodo utilizzato. I valori validi sono: `DELETE`, `GET`, `HEAD`, `OPTIONS`, `PATCH`, `POST` e `PUT`.
`$context.identity.accountId`	L'ID AWS dell'account associato alla richiesta.
`$context.identity.apiKey`	Per API i metodi che richiedono una API chiave, questa variabile è la API chiave associata alla richiesta del metodo. Per i metodi che non richiedono una API chiave, questa variabile è nulla. Per ulteriori informazioni, consulta the section called "Piani di utilizzo".
`$context.identity.apiKeyId`	L'ID della API chiave associato a una API richiesta che richiede una API chiave.
`$context.identity.caller`	Identificatore dell'entità principale dell'intermediario che ha firmato la richiesta. Supportato per le risorse che utilizzano IAM l'autorizzazione.

Parametro	Descrizione
`$context.identity.cognitoAuthenticationProvider`	Un elenco separato da virgole di tutti i provider di autenticazione Amazon Cognito utilizzati dal chiamante che effettua la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito. Ad esempio, per un'identità di un pool di utenti Amazon Cognito, `cognito-idp.`*`region`*`.amazonaws.com/`*`user_pool_id`*`,cognito-idp.`*`region`*`.amazonaws.com/`*`user_pool_id`*`:CognitoSignIn:`*`token subject claim`* Per informazioni sui provider di autenticazione di Amazon Cognito disponibili, consulta Using Federated Identities nella Amazon Cognito Developer Guide.
`$context.identity.cognitoAuthenticationType`	Tipo di autenticazione Amazon Cognito dell'intermediario da cui proviene la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito. I valori possibili includono `authenticated` per le identità autenticate e `unauthenticated` per le identità non autenticate.
`$context.identity.cognitoIdentityId`	ID identità di Amazon Cognito dell'intermediario da cui proviene la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito.
`$context.identity.cognitoIdentityPoolId`	ID pool di identità di Amazon Cognito dell'intermediario da cui proviene la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito.

Parametro	Descrizione
`$context.identity.principalOrgId`	L'ID organizzazione AWS.
`$context.identity.sourceIp`	L'indirizzo IP di origine della TCP connessione immediata che effettua la richiesta all'endpoint Gateway. API
`$context.identity.clientCert.clientCertPem`	Il certificato client PEM con codifica che il client ha presentato durante l'autenticazione reciproca. TLS Presente quando un client accede a un API utilizzando un nome di dominio personalizzato con attivazione reciproca. TLS Presente solo nei log di accesso se l'TLSautenticazione reciproca fallisce.
`$context.identity.clientCert.subjectDN`	Nome distinto dell'oggetto del certificato presentato da un client. Presente quando un client accede a un utente API utilizzando un nome di dominio personalizzato con mutua TLS attivazione. Presente solo nei log di accesso se l'TLSautenticazione reciproca fallisce.
`$context.identity.clientCert.issuerDN`	Nome distinto dell'approvatore del certificato presentato da un cliente. Presente quando un client accede a un utente API utilizzando un nome di dominio personalizzato con mutua TLS attivazione. Presente solo nei log di accesso se l'TLSautenticazione reciproca fallisce.
`$context.identity.clientCert.serialNumber`	Il numero di serie del certificato. Presente quando un client accede a un utente API utilizzando un nome di dominio personalizzato con mutua TLS attivazione. Presente solo nei log di accesso se l'TLSautenticazione reciproca fallisce.

Parametro	Descrizione
`$context.identity.clientCert.validity.notBefore`	La data prima della quale il certificato non è valido. Presente quando un client accede a un utente API utilizzando un nome di dominio personalizzato con mutua TLS attivazione. Presente solo nei log di accesso se l'TLSautenticazione reciproca fallisce.
`$context.identity.clientCert.validity.notAfter`	La data dopo la quale il certificato non è valido. Presente quando un client accede a un utente API utilizzando un nome di dominio personalizzato con mutua TLS attivazione. Presente solo nei log di accesso se l'TLSautenticazione reciproca fallisce.
`$context.identity.vpcId`	L'VPCID di chi VPC effettua la richiesta all'endpoint API Gateway.
`$context.identity.vpceId`	L'ID dell'VPCendpoint che effettua la richiesta all'VPCendpoint Gateway. API Presente solo quando ne hai uno privato. API
`$context.identity.user`	Identificatore dell'entità principale dell'utente che sarà autorizzato per l'accesso alle risorse. Supportato per le risorse che utilizzano IAM l'autorizzazione.
`$context.identity.userAgent`	L'`User-Agent` intestazione del API chiamante .
`$context.identity.userArn`	L'Amazon Resource Name (ARN) dell'utente effettivo identificato dopo l'autenticazione. Per ulteriori informazioni, consulta https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html.

Parametro	Descrizione
`$context.isCanaryRequest`	Indica `true` se la richiesta è stata indirizzata al canarino e `false` se la richiesta non è stata indirizzata al canarino. Presente solo quando hai un canarino abilitato.
`$context.path`	Percorso della richiesta. Ad esempio, per una richiesta non proxy URL di`https://{rest-api-id}.execute-api.{region}.amazonaws.com/{stage}/root/child`, il `$context.path` valore è. `/{stage}/root/child`
`$context.protocol`	Protocollo della richiesta, ad esempi, `HTTP/1.1`. **Note** APIGateway APIs può accettare richieste HTTP /2, ma API Gateway invia richieste alle integrazioni di backend utilizzando /1.1. HTTP Di conseguenza, il protocollo di richiesta viene registrato come HTTP /1.1 anche se un client invia una richiesta che utilizza /2. HTTP
`$context.requestId`	L'ID della richiesta. I client possono sovrascrivere questo ID di richiesta. Utilizzalo `$context.extendedRequestId` per un ID di richiesta univoco generato da GatewayAPI.

Parametro	Descrizione
`$context.requestOverride.header.` *`header_name`*	Sovrascrittura intestazione della richiesta. Se questo parametro è definito, contiene le intestazioni da utilizzare al posto delle HTTPintestazioni definite nel riquadro Integration Request. Per ulteriori informazioni, consulta Utilizza un modello di mappatura per sovrascrivere i parametri API di richiesta e risposta e i codici di stato di un.
`$context.requestOverride.path.` *`path_name`*	Sovrascrittura percorso della richiesta. Se questo parametro è definito, contiene il percorso della richiesta da utilizzare anziché i parametri del URL percorso definiti nel riquadro Richiesta di integrazione. Per ulteriori informazioni, consulta Utilizza un modello di mappatura per sovrascrivere i parametri API di richiesta e risposta e i codici di stato di un.
`$context.requestOverride.querystring.` *`querystring_name`*	Sovrascrittura stringa di query della richiesta. Se questo parametro è definito, contiene le stringhe di query di richiesta da utilizzare al posto dei parametri della stringa di URL query definiti nel riquadro Richiesta di integrazione. Per ulteriori informazioni, consulta Utilizza un modello di mappatura per sovrascrivere i parametri API di richiesta e risposta e i codici di stato di un.

Parametro	Descrizione
`$context.responseOverride.header.` *`header_name`*	Sovrascrittura intestazione della risposta. Se definito, questo parametro contiene l'intestazione da restituire al posto della Response header (Intestazione di risposta) che è definita come la Default mapping (mappatura predefinita) nel riquadro Integration Response (Risposta integrazione). Per ulteriori informazioni, consulta Utilizza un modello di mappatura per sovrascrivere i parametri API di richiesta e risposta e i codici di stato di un.
`$context.responseOverride.status`	Sovrascrittura codice di stato della risposta. Se definito, questo parametro contiene il codice di stato da restituire al posto di Method response status (Stato risposta metodo) che è definito come la Default mapping (Mappatura predefinita) nel riquadro Integration Response (Risposta integrazione). Per ulteriori informazioni, consulta Utilizza un modello di mappatura per sovrascrivere i parametri API di richiesta e risposta e i codici di stato di un.
`$context.requestTime`	L'ora della richiesta in CLFformato -formattato (). `dd/MMM/yyyy:HH:mm:ss +-hhmm`
`$context.requestTimeEpoch`	L'ora della richiesta in formato epoca (Unix epoch) in millisecondi.
`$context.resourceId`	L'identificatore che API Gateway assegna alla tua risorsa.

Parametro	Descrizione
`$context.resourcePath`	Percorso della risorsa. Ad esempio, per la richiesta non proxy URI di`https://{rest-api-id}.execute-api.{region}.amazonaws.com/{stage}/root/child` , Il `$context.resourcePath` valore è. `/root/child` Per ulteriori informazioni, consulta Tutorial: creare un file REST API con un'HTTPintegrazione non proxy.
`$context.stage`	La fase di distribuzione della API richiesta (ad esempio, `Beta` o`Prod`).
`$context.wafResponseCode`	La risposta ricevuta da AWS WAF: `WAF_ALLOW` o `WAF_BLOCK` . Non verrà impostato se lo stage non è associato a un WebACL. Per ulteriori informazioni, consulta the section called “AWS WAF”.
`$context.webaclArn`	Il sito Web ACL completo ARN utilizzato per decidere se consentire o bloccare la richiesta . Non verrà impostato se lo stage non è associato a un webACL. Per ulteriori informazioni, consulta the section called “AWS WAF”.

Esempio di modello di variabile **`$context`**

Potresti voler utilizzare `$context` le variabili in un modello di mappatura se il tuo API metodo passa dati strutturati a un backend che richiede che i dati siano in un formato particolare.

L'esempio seguente mostra un modello di mappatura che mappa variabili `$context` in ingresso a variabili back-end con nomi leggermente diversi nel payload di una richiesta di integrazione:

> **Note**
>
> Una delle variabili è una API chiave. L'esempio presuppone che il metodo richieda una API chiave.

```
{
    "stage" : "$context.stage",
    "request_id" : "$context.requestId",
    "api_id" : "$context.apiId",
    "resource_path" : "$context.resourcePath",
    "resource_id" : "$context.resourceId",
    "http_method" : "$context.httpMethod",
    "source_ip" : "$context.identity.sourceIp",
    "user-agent" : "$context.identity.userAgent",
    "account_id" : "$context.identity.accountId",
    "api_key" : "$context.identity.apiKey",
    "caller" : "$context.identity.caller",
    "user" : "$context.identity.user",
    "user_arn" : "$context.identity.userArn"
}
```

L'output di questo modello di mappatura dovrebbe essere simile al seguente:

```
{
  stage: 'prod',
  request_id: 'abcdefg-000-000-0000-abcdefg',
  api_id: 'abcd1234',
  resource_path: '/',
  resource_id: 'efg567',
  http_method: 'GET',
  source_ip: '192.0.2.1',
  user-agent: 'curl/7.84.0',
  account_id: '111122223333',
  api_key: 'MyTestKey',
  caller: 'ABCD-0000-12345',
  user: 'ABCD-0000-12345',
  user_arn: 'arn:aws:sts::111122223333:assumed-role/Admin/carlos-salazar'
}
```

Variabili **`$context`** solo per la registrazione degli accessi

Le seguenti variabili `$context` sono disponibili solo per la registrazione degli accessi. Per ulteriori informazioni, consulta the section called “CloudWatch registri”. (Per WebSocket APIs, vedithe section called “Metriche”.)

Parametro	Descrizione
`$context.authorize.error`	Il messaggio di errore di autorizzazione.
`$context.authorize.latency`	La latenza di autorizzazione in ms.
`$context.authorize.status`	Il codice di stato restituito da un tentativo di autorizzazione.
`$context.authorizer.error`	Il messaggio di errore restituito da un'autori zzazione.
`$context.authorizer.integra tionLatency`	La latenza di integrazione dell'autorizzatore in ms.
`$context.authorizer.integra tionStatus`	Il codice di stato restituito da un'autorizzazione Lambda.
`$context.authorizer.latency`	La latenza di autorizzazione in ms.
`$context.authorizer.requestId`	L'ID della AWS richiesta dell'endpoint.
`$context.authorizer.status`	Il codice di stato restituito da un'autorizzazione.
`$context.authenticate.error`	Il messaggio di errore restituito da un tentativo di autenticazione.
`$context.authenticate.latency`	La latenza di autenticazione in ms.
`$context.authenticate.status`	Il codice di stato restituito da un tentativo di autenticazione.
`$context.customDomain.baseP athMatched`	Il percorso di una API mappatura a cui corrisponde una richiesta in arrivo. Applicabile quando un client utilizza un nome di dominio

Parametro	Descrizione
	personalizzato per accedere a un. API Ad esempio, se un client invia una richiesta `ahttps://api.example.com/v1/orders/1234`, e la richiesta corrispon de alla API mappatura con il percorso`v1/orders`, il valore è`v1/orders`. Per ulteriori informazioni, consulta the section called "APImappature".
`$context.endpointType`	Il tipo di endpoint di. API
`$context.integration.error`	Il messaggio di errore restituito da un'integr azione.
`$context.integration.integrationStatus`	Per l'integrazione del proxy Lambda, il codice di stato restituito dal codice della funzione Lambda di backend AWS Lambda, non dal codice della funzione Lambda.
`$context.integration.latency`	Latenza di integrazione in ms. Equivalente a `$context.integrationLatency`.
`$context.integration.requestId`	L'ID della AWS richiesta dell'endpoint. Equivalente a `$context.awsEndpointRequestId`.
`$context.integration.status`	Il codice di stato restituito da un'integrazione. Per le integrazioni proxy Lambda, questo è il codice di stato restituito dal codice della funzione Lambda.
`$context.integrationLatency`	Latenza di integrazione in ms.

Parametro	Descrizione
`$context.integrationStatus`	Per l'integrazione del proxy Lambda, questo parametro rappresenta il codice di stato restituit o dal codice della funzione Lambda di AWS Lambda backend e non dal codice della funzione Lambda.
`$context.responseLatency`	Latenza della risposta in ms.
`$context.responseLength`	Lunghezza del payload della risposta in byte.
`$context.status`	Stato della risposta del metodo.
`$context.waf.error`	Il messaggio di errore restituito da. AWS WAF
`$context.waf.latency`	La AWS WAF latenza in ms.
`$context.waf.status`	Il codice di stato restituito da AWS WAF.
`$context.xrayTraceId`	L'ID della traccia di X-Ray. Per ulteriori informazioni, consulta the section called “Configurare AWS X-Ray”.

`$input` Variabili

La variabile `$input` rappresenta il payload della richiesta del metodo e i parametri che devono essere elaborati dal modello di mappatura. Fornisce le seguenti funzioni:

Variabile e funzione	Descrizione
`$input.body`	Restituisce il payload della richiesta non elaborata come stringa. È possibile utilizzare `$input.body` per conservare interi numeri in virgola mobile, ad esempio`10.00`.
`$input.json(x)`	Questa funzione valuta un'JSONPathespressione e restituisce i risultati sotto forma di stringaJSON.

Variabile e funzione	Descrizione
	Ad esempio, `$input.json('$.pets')` restituisce una JSON stringa che rappresenta la `pets` struttura. Per ulteriori informazioni suJSONPath, vedere JSONPathor JSONPathfor Java.
`$input.params()`	Restituisce una mappa di tutti i parametri della richiesta. Si consiglia di utilizzare `$util.escapeJavaScript` per sanificare il risultato ed evitare un potenziale attacco di iniezione. Per il pieno controllo della sanificazione delle richieste, utilizzare un'integrazione proxy senza un modello e gestire la sanificazione delle richieste nell'integrazione.
`$input.params(x)`	Restituisce il valore di un parametro della richiesta del metodo dal percorso, dalla stringa di query o dal valore dell'intestazione (cercati in questo ordine) a partire da una stringa del nome del parametro `x`. Si consiglia di utilizzare `$util.escapeJavaScript` per sanificare il parametro ed evitare un potenziale attacco di iniezione. Per il controllo completo della sanificazione dei parametri, utilizzare un'integrazione proxy senza un modello e gestire la sanificazione delle richieste nell'integrazione.

Variabile e funzione	Descrizione
`$input.path(x)`	Accetta una stringa di JSONPath espressio ne (x) e restituisce una rappresentazione JSON oggettuale del risultato. Ciò consente di accedere e manipolare gli elementi del payload in modo nativo in Apache Velocity Template Language (). VTL Ad esempio, se l'espressione `$input.path('$.pets')` restituisce un oggetto in questo modo: `[ { "id": 1, "type": "dog", "price": 249.99 }, { "id": 2, "type": "cat", "price": 124.99 }, { "id": 3, "type": "fish", "price": 0.99 } ]` `$input.path('$.pets').size()` restituisce `"3"`. Per ulteriori informazioni suJSONPath, vedere or for Java. JSONPathJSONPath

Esempi di modello di variabile **`$input`**

Gli esempi seguenti mostrano come utilizzare `$input` le variabili nei modelli di mappatura. Puoi utilizzare un'integrazione fittizia o un'integrazione non proxy Lambda che restituisca l'evento di input API a Gateway per provare questi esempi.

Esempio di modello di mappatura di parametri

L'esempio seguente passa tutti i parametri della richiesta, inclusi`path`, e `querystringheader`, all'endpoint di integrazione tramite un payload: JSON

```
#set($allParams = $input.params())
{
  "params" : {
    #foreach($type in $allParams.keySet())
    #set($params = $allParams.get($type))
    "$type" : {
      #foreach($paramName in $params.keySet())
      "$paramName" : "$util.escapeJavaScript($params.get($paramName))"
      #if($foreach.hasNext),#end
      #end
    }
    #if($foreach.hasNext),#end
    #end
  }
}
```

Per una richiesta che include i seguenti parametri di input:

- Un parametro di percorso denominato `myparam`
- Parametri della stringa di query `querystring1=value1,value2&querystring2=value3`
- Intestazioni `"header1" : "value1""header2" : "value2","header3" : "value3"`.

L'output di questo modello di mappatura dovrebbe essere simile al seguente:

```
{
  "params" : {
        "path" : {
            "path" : "myparam"
                }
```

```
    ,        "querystring" : {
            "querystring1" : "value1,value2"
      ,            "querystring2" : "value3"
                }
    ,        "header" : {
            "header3" : "value3"
      ,            "header2" : "value2"
      ,            "header1" : "value1"
                }
          }
}
```

JSONesempio di modello di mappatura

È possibile usare la variabile `$input` per ottenere stringhe di query e il corpo della richiesta con o senza l'uso di modelli. Potresti anche voler ottenere il parametro e il payload o una sottosezione del payload. I tre esempi seguenti mostrano come eseguire questa operazione.

L'esempio seguente utilizza un modello di mappatura per ottenere una sottosezione del payload. Questo esempio ottiene il parametro di input `name` e quindi l'intero corpo: POST

```
{
    "name" : "$input.params('name')",
    "body" : $input.json('$')
}
```

Per una richiesta che include i parametri della stringa di query `name=Bella&type=dog` e il seguente corpo:

```
{
    "Price" : "249.99",
    "Age": "6"
}
```

L'output di questo modello di mappatura dovrebbe essere simile al seguente:

```
{
    "name" : "Bella",
    "body" : {"Price":"249.99","Age":"6"}
}
```

Se l'JSONinput contiene caratteri senza escape che non possono essere analizzati JavaScript, API Gateway potrebbe restituire una risposta 400. Applica `$util.escapeJavaScript($input.json('$'))` per garantire che l'JSONinput possa essere analizzato correttamente.

L'esempio precedente con `$util.escapeJavaScript($input.json('$'))` applied è il seguente:

```
{
    "name" : "$input.params('name')",
    "body" : $util.escapeJavaScript($input.json('$'))
}
```

In questo caso, l'output di questo modello di mappatura dovrebbe essere simile al seguente:

```
{
    "name" : "Bella",
    "body": {\"Price\":\"249.99\",\"Age\":\"6\"}
}
```

JSONPathesempio di espressione

L'esempio seguente mostra come passare un'JSONPathespressione al `json()` metodo. Puoi anche leggere una sottosezione dell'oggetto del corpo della richiesta utilizzando un punto, `.`, per specificare una proprietà:

```
{
    "name" : "$input.params('name')",
    "body" : $input.json('$.Age')
}
```

Per una richiesta che include i parametri della stringa di query `name=Bella&type=dog` e il seguente corpo:

```
{
    "Price" : "249.99",
    "Age": "6"
}
```

L'output di questo modello di mappatura dovrebbe essere simile al seguente:

```
{
    "name" : "Bella",
    "body" : "6"
}
```

Se il payload di una richiesta di metodo contiene caratteri senza escape che non possono essere analizzati JavaScript, API Gateway potrebbe restituire una risposta. `400` Applica `$util.escapeJavaScript()` per garantire che l'JSONinput possa essere analizzato correttamente.

L'esempio precedente con `$util.escapeJavaScript($input.json('$.Age'))` applied è il seguente:

```
{
    "name" : "$input.params('name')",
    "body" : "$util.escapeJavaScript($input.json('$.Age'))"
}
```

In questo caso, l'output di questo modello di mappatura dovrebbe essere simile al seguente:

```
{
    "name" : "Bella",
    "body": "\"6\""
}
```

Esempio di richiesta e risposta

L'esempio seguente utilizza `$input.params()`$input.path(), e `$input.json()` per una risorsa con il percorso`/things/{id}`:

```
{
    "id" : "$input.params('id')",
    "count" : "$input.path('$.things').size()",
    "things" : $input.json('$.things')"
}
```

Per una richiesta che include il parametro path `123` e il seguente corpo:

```
{
      "things": {
            "1": {},
```

```
            "2": {},
            "3": {}
      }
}
```

L'output di questo modello di mappatura dovrebbe essere simile al seguente:

```
{"id":"123","count":"3","things":{"1":{},"2":{},"3":{}}}
```

Se il payload di una richiesta di metodo contiene caratteri senza escape che non possono essere analizzati JavaScript, API Gateway potrebbe restituire una risposta. `400` Applica `$util.escapeJavaScript()` per garantire che l'JSONinput possa essere analizzato correttamente.

L'esempio precedente con `$util.escapeJavaScript($input.json('$.things'))` applied è il seguente:

```
{
     "id" : "$input.params('id')",
     "count" : "$input.path('$.things').size()",
     "things" : "$util.escapeJavaScript($input.json('$.things'))"
}
```

L'output di questo modello di mappatura dovrebbe essere simile al seguente:

```
{"id":"123","count":"3","things":"{\"1\":{},\"2\":{},\"3\":{}}"}
```

Per ulteriori esempi di mappatura, consulta Modelli di mappatura per REST APIs.

`$stageVariables`

Le variabili di fase possono essere utilizzate nei modelli di mappatura e mappatura dei parametri e come segnaposto ARNs e URLs utilizzate nelle integrazioni di metodi. Per ulteriori informazioni, consulta the section called “Utilizzo delle variabili di fase”.

Sintassi	Descrizione
`$stageVariables.` *`<variable_name>`* , `$stageVariables['` *`<variable`*	*`<variable_name>`* rappresenta il nome di una variabile di fase.

Sintassi	Descrizione
`_name>` `']` o `${stageVariables['` `<variable_name>` `']}`	

`$util` Variabili

La variabile `$util` contiene funzioni di utilità da usare in modelli di mappatura.

> **Note**
>
> Se non diversamente specificato, il set di caratteri predefinito è UTF -8.

Funzione	Descrizione
`$util.escapeJavaScript()`	Sfugge ai caratteri di una stringa utilizzando le regole delle JavaScript stringhe. **Note** Questa funzione trasforma qualsiasi virgoletta singola (`'`) in virgoletta preceduta da un carattere escape (`\'`). Tuttavia, le virgolette singole con escape non sono valide in. JSON Pertanto, quando l'output di questa funzione viene utilizzato in una JSON proprietà, è necessario riconvertire le virgolette singole con escape (`\'`) in virgolette singole regolari (`'`). Questo viene mostrato nell'esempio seguente: `"input" : "$util.escapeJavaScript( data).replaceAll("\\'","'")"`

Funzione	Descrizione

`$util.parseJson()`	Prende «stringified» JSON e restituisce una rappresentazione oggettuale del risultato. È possibile utilizzare il risultato di questa funzione per accedere e manipolare gli elementi del payload in modo nativo in Apache Velocity Template Language (). VTL Ad esempio, in presenza del payload seguente:
	`{"errorMessage":"{\"key1\":\"var1\", \"key2\":{\"arr\":[1,2,3]}}"}`
	E se usi il modello di mappatura seguente:
	`#set ($errorMessageObj = $util.parseJson($input.path('$.errorMessage')))`
	`{`
	`"errorMessageObjKey2ArrVal" : $errorMessageObj.key2.arr[0]`
	`}`
	Otterrai l'output seguente:
	`{`
	`"errorMessageObjKey2ArrVal" : 1`
	`}`

`$util.urlEncode()`	Converte una stringa nel formato «application/». x-www-form-urlencoded
`$util.urlDecode()`	Decodifica una stringa «application/». x-www-form-urlencoded
`$util.base64Encode()`	Codifica i dati in una stringa con codifica base64.

Funzione	Descrizione
`$util.base64Decode()`	Decodifica i dati da una stringa con codifica base64.

Risposte gateway per REST APIs in API Gateway

Una risposta gateway è identificata da un tipo di risposta definito da API Gateway. La risposta è costituita da un codice di HTTP stato, un set di intestazioni aggiuntive specificate mediante mappature dei parametri e un payload generato da un modello non di mappatura. VTL

Nel API Gateway RESTAPI, una risposta del gateway è rappresentata da. GatewayResponse In OpenAPI, un'`GatewayResponse`istanza è descritta dalle x-amazon-apigateway-gateway-response.gatewayResponseestensione.

Per abilitare una risposta gateway, è necessario impostare una risposta gateway per un tipo di risposta supportato a API livello. Ogni volta che API Gateway restituisce una risposta di questo tipo, vengono applicati i modelli di header mapping e payload mapping definiti nella risposta del gateway per restituire i risultati mappati al chiamante. API

Nella sezione seguente, mostriamo come configurare le risposte del gateway utilizzando la console Gateway e il Gateway. API API REST API

Configurazione delle risposte del gateway per la personalizzazione delle risposte agli errori

Se API Gateway non riesce a elaborare una richiesta in entrata, restituisce al client una risposta di errore senza inoltrare la richiesta al backend di integrazione. Per impostazione predefinita, la risposta di errore contiene un breve messaggio di errore descrittivo. Ad esempio, se si tenta di chiamare un'operazione su una API risorsa non definita, si riceve una risposta di errore con il messaggio. `{ "message": "Missing Authentication Token" }` Se non conosci API Gateway, potresti avere difficoltà a capire cosa sia effettivamente andato storto.

Per alcune risposte di errore, API Gateway consente API agli sviluppatori di personalizzare le risposte in diversi formati. Per l'esempio `Missing Authentication Token`, puoi aggiungere un suggerimento al payload originale della risposta con la possibile causa, come in questo esempio: `{"message":"Missing Authentication Token", "hint":"The HTTP method or resources may not be supported."}`.

Quando si API interagisce tra uno scambio esterno e il AWS Cloud, si utilizzano modelli di VTL mappatura per la richiesta di integrazione o la risposta di integrazione per mappare il payload da un formato all'altro. Tuttavia, i modelli di VTL mappatura funzionano solo per richieste valide con risposte riuscite.

Per le richieste non valide, API Gateway ignora completamente l'integrazione e restituisce una risposta di errore. Devi utilizzare la personalizzazione per rendere le risposte di errore in un formato conforme allo scambio. Qui, la personalizzazione viene resa in un modello non VTL mappato che supporta solo semplici sostituzioni di variabili.

Generalizzando la risposta di errore generata dal API gateway a qualsiasi risposta generata da API Gateway, le chiamiamo risposte gateway. Ciò distingue le risposte generate dal API Gateway dalle risposte di integrazione. Un modello di mappatura di una risposta del gateway può accedere ai valori della variabile `$context` e ai valori della proprietà `$stageVariables`, nonché ai parametri della richiesta di metodo, sotto forma di `method.request.`*`param-position`*`.`*`param-name`*.

Per ulteriori informazioni sulle variabili `$context`, consulta `$context`Variabili per modelli di dati, autorizzatori, modelli di mappatura e registrazione degli accessi CloudWatch . Per ulteriori informazioni su `$stageVariables`, consulta `$stageVariables`. Per ulteriori informazioni sui parametri di richiesta del metodo, consulta the section called “`$input` Variabili ”.

Argomenti

- Imposta una risposta gateway per un REST API utilizzo della console Gateway API
- Imposta una risposta del gateway utilizzando il Gateway API REST API
- Configura la personalizzazione della risposta del gateway in Open API
- Tipi di risposta del gateway per API Gateway

Imposta una risposta gateway per un REST API utilizzo della console Gateway API

L'esempio seguente mostra come impostare una risposta gateway per un REST API utilizzo della console API Gateway

Per personalizzare una risposta gateway utilizzando la console API Gateway

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli un RESTAPI.
3. Nel riquadro di navigazione principale di Gateway API, scegli Risposte di Gateway.

4. Scegli un tipo di risposta, quindi scegli Modifica. In questa procedura guidata, utilizziamo Token di autenticazione mancante come esempio.
5. Puoi modificare il codice di stato API generato dal gateway per restituire un codice di stato diverso che soddisfi i tuoi API requisiti. In questo esempio, la personalizzazione cambia il codice di stato da quello predefinito (`403`) al codice `404`, poiché questo messaggio di errore viene ricevuto quando un client chiama una risorsa non valida o non supportata che può essere considerata come non trovata.
6. Per tornare alle intestazioni personalizzate, seleziona Aggiungi intestazione della risposta in Intestazioni di risposta. A scopo illustrativo, aggiungiamo le seguenti intestazioni personalizzate:

   ```
   Access-Control-Allow-Origin:'a.b.c'
   x-request-id:method.request.header.x-amzn-RequestId
   x-request-path:method.request.path.petId
   x-request-query:method.request.querystring.q
   ```

   Nelle mappature di intestazione precedenti, un nome di dominio statico (`'a.b.c'`) viene mappato all'`Allow-Control-Allow-Origin`intestazione per consentire l'CORSaccesso aAPI; l'intestazione della richiesta di input di `x-amzn-RequestId` è mappata `request-id` nella risposta; la variabile `petId` path della richiesta in entrata viene mappata all'`request-path`intestazione nella risposta; e il parametro di q query della richiesta originale è mappato all'`request-query`intestazione della risposta.
7. In Modelli di risposta, non modificare `application/json` in Tipo di contenuto e immetti il seguente modello di mappatura del corpo nell'editor Corpo modello:

   ```
   {
        "message":"$context.error.messageString",
        "type": "$context.error.responseType",
        "statusCode": "'404'",
        "stage": "$context.stage",
        "resourcePath": "$context.resourcePath",
        "stageVariables.a": "$stageVariables.a"
   }
   ```

   Questo modello mostra come mappare le proprietà `$context` e `$stageVariables` alle proprietà del corpo di risposta del gateway.
8. Seleziona Salvataggio delle modifiche.
9. APIDistribuiscili in una fase nuova o esistente.

Verifica la risposta del gateway chiamando il CURL comando seguente, supponendo che la chiamata URL del API metodo corrispondente sia: `https://o81lxisefl.execute-api.us-east-1.amazonaws.com/custErr/pets/{petId}`

```
curl -v -H 'x-amzn-RequestId:123344566' https://o81lxisefl.execute-api.us-
east-1.amazonaws.com/custErr/pets/5/type?q=1
```

Poiché il parametro aggiuntivo della stringa di query q=1 non è compatibile conAPI, viene restituito un errore dalla risposta del gateway specificata. Dovresti visualizzare una risposta del gateway simile alla seguente:

```
> GET /custErr/pets/5?q=1 HTTP/1.1
Host: o81lxisefl.execute-api.us-east-1.amazonaws.com
User-Agent: curl/7.51.0
Accept: */*

HTTP/1.1 404 Not Found
Content-Type: application/json
Content-Length: 334
Connection: keep-alive
Date: Tue, 02 May 2017 03:15:47 GMT
x-amzn-RequestId: 123344566
Access-Control-Allow-Origin: a.b.c
x-amzn-ErrorType: MissingAuthenticationTokenException
header-1: static
x-request-query: 1
x-request-path: 5
X-Cache: Error from cloudfront
Via: 1.1 441811a054e8d055b893175754efd0c3.cloudfront.net (CloudFront)
X-Amz-Cf-Id: nNDR-fX4csbRoAgtQJ16u0rTDz9FZWT-Mk93KgoxnfzDlTUh3flmzA==

{
     "message":"Missing Authentication Token",
     "type": MISSING_AUTHENTICATION_TOKEN,
     "statusCode": '404',
     "stage": custErr,
     "resourcePath": /pets/{petId},
     "stageVariables.a": a
}
```

L'esempio precedente presuppone che il API backend sia Pet Store e che API abbia una variabile stage,, a defined.

Imposta una risposta del gateway utilizzando il Gateway API REST API

Prima di personalizzare una risposta gateway utilizzando il API Gateway RESTAPI, è necessario aver già creato API e ottenuto il relativo identificatore. Per recuperare l'APIdentificatore, puoi seguire la relazione di collegamento restapi:gateway-response ed esaminare il risultato.

Per personalizzare una risposta del gateway utilizzando il Gateway API REST API

1. Per sovrascrivere un'intera GatewayResponseistanza, chiama il gatewayresponse:put action. Specificate un valore desiderato responseTypenel parametro URL path e fornite nel payload della richiesta le mappature, e. statusCoderesponseParametersresponseTemplates
2. Per aggiornare parte di un'istanza `GatewayResponse`, chiamare l'operazione gatewayresponse:update. Specificate un valore desiderato `responseType` nel parametro URL path e fornite nel payload della richiesta le singole `GatewayResponse` proprietà desiderate, ad esempio la o la mappatura. `responseParameters` `responseTemplates`

Configura la personalizzazione della risposta del gateway in Open API

È possibile utilizzare l'`x-amazon-apigateway-gateway-responses`estensione a livello API principale per personalizzare le risposte del gateway in OpenAPI. La seguente API definizione aperta mostra un esempio di personalizzazione GatewayResponsedel `MISSING_AUTHENTICATION_TOKEN` tipo.

```
  "x-amazon-apigateway-gateway-responses": {
    "MISSING_AUTHENTICATION_TOKEN": {
      "statusCode": 404,
      "responseParameters": {
        "gatewayresponse.header.x-request-path": "method.input.params.petId",
        "gatewayresponse.header.x-request-query": "method.input.params.q",
        "gatewayresponse.header.Access-Control-Allow-Origin": "'a.b.c'",
        "gatewayresponse.header.x-request-header": "method.input.params.Accept"
      },
      "responseTemplates": {
        "application/json": "{\n     \"message\": $context.error.messageString,\n
    \"type\":  \"$context.error.responseType\",\n     \"stage\":  \"$context.stage
\",\n     \"resourcePath\":  \"$context.resourcePath\",\n     \"stageVariables.a\":
 \"$stageVariables.a\",\n     \"statusCode\": \"'404'\"\n}"
      }
    }
```

In questo esempio, la personalizzazione cambia il codice di stato dal predefinito (`403`) a `404`. Aggiunge anche alla risposta del gateway quattro parametri di intestazione e un modello di mappatura del corpo per il tipo di supporto `application/json`.

Tipi di risposta del gateway per API Gateway

APIGateway espone le seguenti risposte gateway per la personalizzazione da parte API degli sviluppatori.

Tipo di risposta del gateway	Codice di stato predefinito	Descrizione
`ACCESS_DENIED`	`403`	Risposta del gateway per un'autorizzazione non riuscita, ad esempio quando l'accesso viene negato da un'autorizzazione ad hoc o di Amazon Cognito. Se il tipo di risposta non è specificato, la risposta viene impostata in modo predefinito sul tipo `DEFAULT_4XX` .
`API_CONFIGURATION_ERROR`	`500`	La risposta del gateway per una API configurazione non valida, ad esempio quando viene inviato un indirizzo endpoint non valido, quando la decodifica base64 non riesce sui dati binari quando viene attivato il supporto binario o quando la mappatura della risposta di integrazione non può corrispondere a nessun modello e non è configurato alcun modello predefinito. Se il tipo di risposta non è specifica to, la risposta viene impostata

Tipo di risposta del gateway	Codice di stato predefinito	Descrizione
		in modo predefinito sul tipo `DEFAULT_5XX` .
`AUTHORIZER_CONFIGURATION_ERROR`	`500`	Risposta del gateway per la mancata connessione ad autorizzazioni ad hoc o di Amazon Cognito. Se il tipo di risposta non è specificato, la risposta viene impostata in modo predefinito sul tipo `DEFAULT_5XX` .
`AUTHORIZER_FAILURE`	`500`	Risposta del gateway quando autorizzazioni ad hoc o di Amazon Cognito non riescono ad autenticare l'intermediario. Se il tipo di risposta non è specificato, la risposta viene impostata in modo predefinito sul tipo `DEFAULT_5XX` .
`BAD_REQUEST_PARAMETERS`	`400`	Riposta del gateway quando il parametro di richiesta non può essere convalidato in base a un validatore abilitato delle richieste. Se il tipo di risposta non è specificato, la risposta viene impostata in modo predefinito sul tipo `DEFAULT_4XX` .

Tipo di risposta del gateway	Codice di stato predefinito	Descrizione
BAD_REQUEST_BODY	400	Riposta del gateway quando il corpo della richiesta non può essere convalidato in base a un validatore abilitato delle richieste. Se il tipo di risposta non è specificato, la risposta viene impostata in modo predefinito sul tipo DEFAULT_4XX .
DEFAULT_4XX	Null	La risposta predefinita del gateway per un tipo di risposta non specificato con codice di stato 4XX. Cambiando il codice di stato di questa risposta di fallback del gateway significa cambiare i codici di stato di tutte le altre risposte 4XX al nuovo valore. Reimpostando il codice di stato su null, i codici di stato di tutte le altre risposte 4XX ritornano ai loro valori originali. **Note** AWS WAF le risposte personalizzate hanno la precedenza sulle risposte gateway personalizzate.

Tipo di risposta del gateway	Codice di stato predefinito	Descrizione
DEFAULT_5XX	Null	La risposta predefinita del gateway per un tipo di risposta non specificato con codice di stato 5XX. Cambiando il codice di stato di questa risposta di fallback del gateway significa cambiare i codici di stato di tutte le altre risposte 5XX al nuovo valore. Reimpostando il codice di stato su null, i codici di stato di tutte le altre risposte 5XX ritornano ai loro valori originali.
EXPIRED_TOKEN	403	La risposta del gateway per un errore del token AWS di autenticazione è scaduta. Se il tipo di risposta non è specifica to, la risposta viene impostata in modo predefinito sul tipo DEFAULT_4XX .
INTEGRATION_FAILURE	504	Risposta del gateway per un errore di integrazione non riuscita. Se il tipo di risposta non è specificato, la risposta viene impostata in modo predefinito sul tipo DEFAULT_5XX .

Tipo di risposta del gateway	Codice di stato predefinito	Descrizione
`INTEGRATION_TIMEOUT`	`504`	Risposta del gateway per un errore di integrazione scaduta. Se il tipo di risposta non è specificato, la risposta viene impostata in modo predefinito sul tipo `DEFAULT_5XX` .
`INVALID_API_KEY`	`403`	La risposta del gateway per una API chiave non valida inviata per un metodo che richiede una API chiave. Se il tipo di risposta non è specifica to, la risposta viene impostata in modo predefinito sul tipo `DEFAULT_4XX` .
`INVALID_SIGNATURE`	`403`	La risposta del gateway per un errore di AWS firma non valido. Se il tipo di risposta non è specificato, la risposta viene impostata in modo predefinito sul tipo `DEFAULT_4XX` .
`MISSING_AUTHENTICA TION_TOKEN`	`403`	La risposta del gateway per un errore del token di autentica zione mancante, inclusi i casi in cui il client tenta di richiamar e un API metodo o una risorsa non supportati. Se il tipo di risposta non è specificato, la risposta viene impostata in modo predefinito sul tipo `DEFAULT_4XX` .

Tipo di risposta del gateway	Codice di stato predefinito	Descrizione
`QUOTA_EXCEEDED`	`429`	Risposta del gateway per un errore di superamento della quota di utilizzo del piano. Se il tipo di risposta non è specificato, la risposta viene impostata in modo predefinito sul tipo `DEFAULT_4XX` .
`REQUEST_TOO_LARGE`	`413`	Risposta del gateway per un errore di dimensioni eccessive della richiesta. Se il tipo di risposta non è specificato, il valore predefini to di questa risposta è:. `HTTP content length exceeded 10485760 bytes`
`RESOURCE_NOT_FOUND`	`404`	La risposta del API gateway quando Gateway non riesce a trovare la risorsa specifica ta dopo che una API richiesta ha superato l'autenticazione e l'autorizzazione, ad eccezione dell'autenticazione e dell'auto rizzazione API delle chiavi. Se il tipo di risposta non è specificato, la risposta viene impostata in modo predefinito sul tipo `DEFAULT_4XX` .

Tipo di risposta del gateway	Codice di stato predefinito	Descrizione
THROTTLED	429	Risposta del gateway quando vengono superati i limiti di throttling a livello di account, fase, metodo o piano. Se il tipo di risposta non è specificato, la risposta viene impostata in modo predefinito sul tipo DEFAULT_4XX .
UNAUTHORIZED	401	Risposta del gateway quando l'autorizzazione ad hoc o di Amazon Cognito non riesce ad autenticare l'intermediario.
UNSUPPORTED_MEDIA_TYPE	415	Risposta del gateway quando il payload presenta un tipo di supporto non supportato, se è stato abilitato un comportamento passthrough severo. Se il tipo di risposta non è specificato, la risposta viene impostata in modo predefinito sul tipo DEFAULT_4XX .

Tipo di risposta del gateway	Codice di stato predefinito	Descrizione
`WAF_FILTERED`	`403`	La risposta del gateway quando una richiesta è bloccato da AWS WAF. Se il tipo di risposta non è specifica to, la risposta viene impostata in modo predefinito sul tipo `DEFAULT_4XX` . **Note** AWS WAF le risposte personalizzate hanno la precedenza sulle risposte gateway personalizzate.

CORSper REST APIs in API Gateway

La condivisione di risorse tra origini (CORS) è una funzionalità di sicurezza del browser che limita HTTP le richieste tra origini diverse avviate da script in esecuzione nel browser. Per ulteriori informazioni, consulta What is? CORS .

Determinare se abilitare CORS il supporto

Una HTTP richiesta multiorigine è quella che viene effettuata per:

- Un dominio diverso (ad esempio da `example.com` a `amazondomains.com`)
- Un sottodominio diverso (ad esempio da `example.com` a `petstore.example.com`)
- Una porta diversa (ad esempio da `example.com` a `example.com:10777`)
- Un protocollo diverso (ad esempio da `https://example.com` a `http://example.com`)

Se non riesci ad accedere al tuo API e ricevi un messaggio di errore che contiene`Cross-Origin Request Blocked`, potrebbe essere necessario abilitarlo. CORS

HTTPLe richieste provenienti da più origini possono essere suddivise in due tipi: richieste semplici e richieste non semplici.

Abilitazione CORS per una richiesta semplice

Una HTTP richiesta è semplice se tutte le seguenti condizioni sono vere:

- Viene emessa contro una API risorsa che consente solo `GET` e `POST` richieste. `HEAD`
- Se si tratta di una richiesta di metodo `POST`, deve includere un'intestazione `Origin`.
- Il tipo di contenuto del payload della richiesta è `text/plain`, `multipart/form-data` o `application/x-www-form-urlencoded`.
- La richiesta non contiene intestazioni personalizzate.
- Eventuali requisiti aggiuntivi elencati nella CORSdocumentazione di Mozilla per richieste semplici.

Per richieste di metodo `POST` multiorigine semplici, la risposta della risorsa deve includere l'intestazione `Access-Control-Allow-Origin: '*'` o `Access-Control-Allow-Origin:'origin'`.

Tutte le altre richieste provenienti da più origini sono HTTP richieste non semplici.

Attivazione di una CORS richiesta non semplice

Se le tue API risorse ricevono richieste non semplici, devi abilitare un CORS supporto aggiuntivo a seconda del tipo di integrazione.

Abilitazione CORS per integrazioni non proxy

Per queste integrazioni, il CORSprotocollo richiede che il browser invii una richiesta di preflight al server e attenda l'approvazione (o una richiesta di credenziali) dal server prima di inviare la richiesta effettiva. È necessario configurare l'invio API di una risposta appropriata alla richiesta di preflight.

Creazione di una risposta di verifica preliminare

1. Creare un metodo `OPTIONS` per l'integrazione fittizia.
2. Aggiungere le seguenti intestazioni di risposta alla risposta del metodo 200:
   - `Access-Control-Allow-Headers`
   - `Access-Control-Allow-Methods`

- `Access-Control-Allow-Origin`

3. Imposta il comportamento del passthrough di integrazione su. `NEVER` In questo caso, la richiesta del metodo di un tipo di contenuto non mappato verrà rifiutata con una risposta HTTP 415 Unsupported Media Type. Per ulteriori informazioni, consulta Comportamenti passthrough di integrazione.
4. Immettere i valori per le intestazioni delle risposte. Per consentire tutte le origini, tutti i metodi e le intestazioni comuni, usare i seguenti valori di intestazione:

   - `Access-Control-Allow-Headers: 'Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token'`
   - `Access-Control-Allow-Methods: '*'`
   - `Access-Control-Allow-Origin: '*'`

Dopo aver creato la richiesta di preflight, è necessario restituire l'`Access-Control-Allow-Origin:` *`'origin'`* intestazione `Access-Control-Allow-Origin: '*'` o per tutti i metodi CORS -enabled per almeno tutte le 200 risposte.

Abilitazione delle CORS integrazioni non proxy utilizzando il AWS Management Console

È possibile utilizzare AWS Management Console per abilitare. CORS APIGateway crea un `OPTIONS` metodo e aggiunge l'`Access-Control-Allow-Origin`intestazione alle risposte di integrazione dei metodi esistenti. Questo non sempre funziona e a volte è necessario modificare manualmente la risposta di integrazione per restituire l'`Access-Control-Allow-Origin`intestazione per tutti i metodi CORS abilitati per almeno tutte le 200 risposte.

Abilitazione del CORS supporto per le integrazioni proxy

Per un'integrazione proxy Lambda o un'integrazione HTTP proxy, il backend è responsabile della restituzione delle `Access-Control-Allow-Headers` intestazioni `Access-Control-Allow-Origin``Access-Control-Allow-Methods`, e, perché un'integrazione proxy non restituisce una risposta di integrazione.

L'esempio seguente di funzioni Lambda restituisce le intestazioni richiesteCORS:

Node.js

```
export const handler = async (event) => {
```

```
    const response = {
        statusCode: 200,
        headers: {
            "Access-Control-Allow-Headers" : "Content-Type",
            "Access-Control-Allow-Origin": "https://www.example.com",
            "Access-Control-Allow-Methods": "OPTIONS,POST,GET"
        },
        body: JSON.stringify('Hello from Lambda!'),
    };
    return response;
};
```

Python 3

```
import json

def lambda_handler(event, context):
    return {
        'statusCode': 200,
        'headers': {
            'Access-Control-Allow-Headers': 'Content-Type',
            'Access-Control-Allow-Origin': 'https://www.example.com',
            'Access-Control-Allow-Methods': 'OPTIONS,POST,GET'
        },
        'body': json.dumps('Hello from Lambda!')
    }
```

Argomenti

- Abilita CORS su una risorsa utilizzando la console Gateway API
- Abilita CORS su una risorsa utilizzando l'importazione da API Gateway API
- Test CORS per un API gateway API

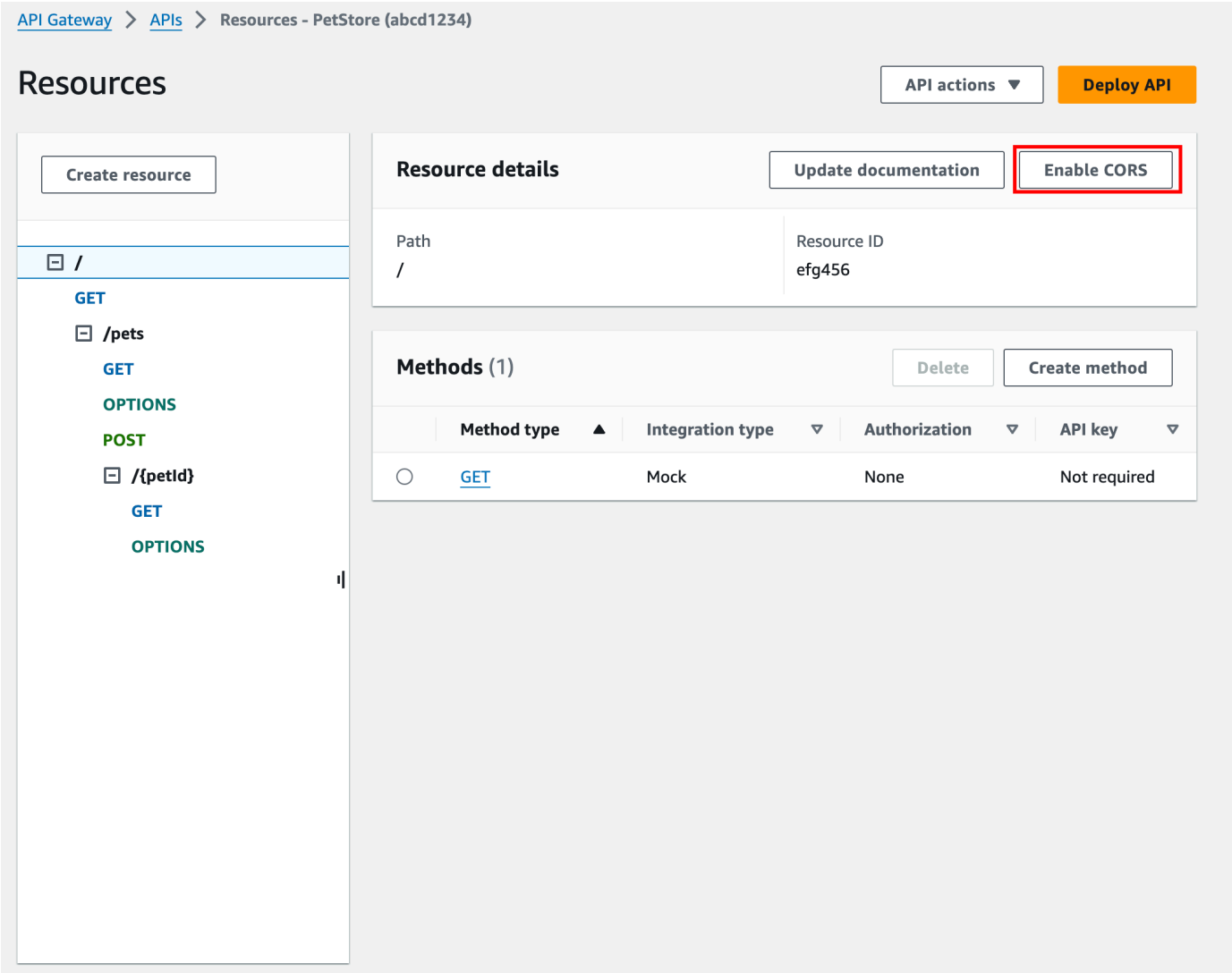
Abilita CORS su una risorsa utilizzando la console Gateway API

È possibile utilizzare la console API Gateway per abilitare il CORS supporto per uno o tutti i metodi su una REST API risorsa creata. Dopo aver abilitato COR il supporto, imposta il comportamento di integrazione passthrough su`NEVER`. In questo caso, la richiesta del metodo di un tipo di contenuto non mappato verrà rifiutata con una risposta HTTP 415 Unsupported Media Type. Per ulteriori informazioni, consulta Comportamenti passthrough di integrazione

> **Important**
>
> Le risorse possono contenere risorse figlio. L'abilitazione del CORS supporto per una risorsa e i relativi metodi non lo abilita in modo ricorsivo per le risorse secondarie e i relativi metodi.

Per abilitare il CORS supporto su una risorsa REST API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere un API.
3. Scegliere una risorsa in Resources (Risorse).
4. Nella sezione Dettagli delle risorse, scegli Abilita CORS.



5. Nella CORS casella Abilita, procedi come segue:

   a. (Facoltativo) Se avete creato una risposta gateway personalizzata e desiderate abilitare CORS il supporto per una risposta, selezionate una risposta gateway.

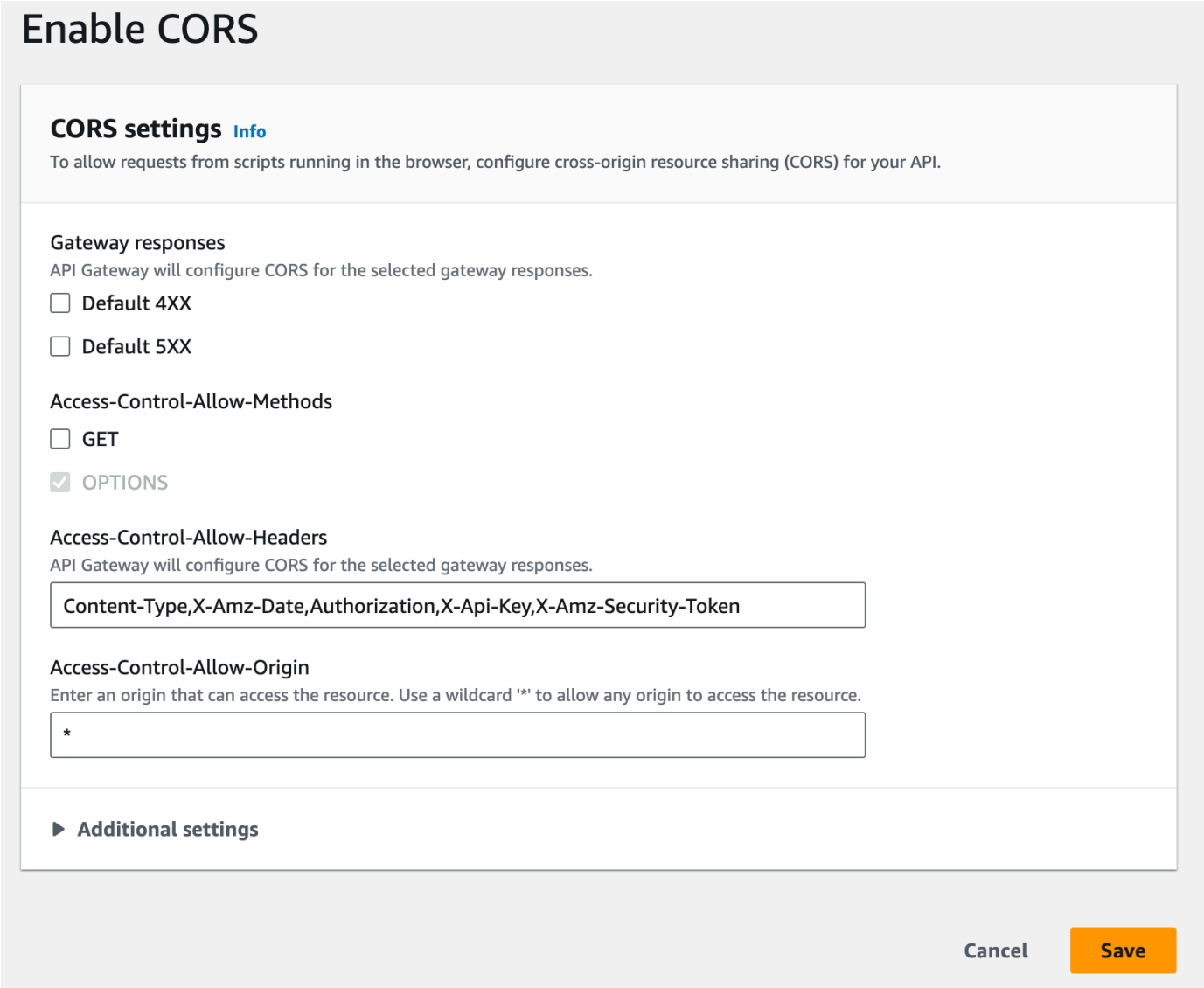
   b. Seleziona ogni metodo per abilitare CORS il supporto. Il `OPTION` metodo deve essere CORS abilitato.

      Se abiliti CORS il supporto per un `ANY` metodo, CORS viene abilitato per tutti i metodi.

   c. Nel campo di input Access-Control-Allow-Headers, immetti una stringa statica di un elenco di intestazioni separate da virgole che il client deve inviare nella richiesta della risorsa. Usa l'elenco di intestazioni `'Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token'` fornito dalla console o specifica intestazioni personalizzate.

   d. Utilizzare il valore `'*'` fornito dalla console come valore dell'intestazione Access-Control-Allow-Origin (Controllo accessi - Autorizza origine) per permettere le richieste di accesso da tutte le origini oppure specificare le origini autorizzate ad accedere alla risorsa.

   e. Selezionare Salva.



> **Important**
>
> Quando si applicano le istruzioni precedenti al `ANY` metodo in un'integrazione proxy, le CORS intestazioni applicabili non verranno impostate. Invece, il backend deve restituire le CORS intestazioni applicabili, ad esempio. `Access-Control-Allow-Origin`

Dopo aver CORS abilitato il `GET` metodo, viene aggiunto un `OPTIONS` metodo alla risorsa, se non è già presente. La risposta `200` del metodo `OPTIONS` viene configurata automaticamente per restituire le tre intestazioni `Access-Control-Allow-*` per soddisfare gli handshake preliminari. Inoltre,

il metodo effettivo (`GET`) è configurato per impostazione predefinita per restituire l'intestazione `Access-Control-Allow-Origin` anche nella risposta 200. Gli altri tipi di risposta devono essere configurati manualmente in modo da restituire l'intestazione `Access-Control-Allow-Origin'` con '*' oppure origini specifiche, se non si vuole che venga restituito l'errore `Cross-origin access`.

Dopo aver abilitato il CORS supporto sulla risorsa, è necessario distribuirla o ridistribuirla API affinché le nuove impostazioni abbiano effetto. Per ulteriori informazioni, consulta the section called “Crea distribuzione”.

> **Note**
>
> Se non è possibile abilitare il CORS supporto sulla risorsa dopo aver seguito la procedura, si consiglia di confrontare la CORS configurazione con la risorsa di esempio. API `/pets` Per informazioni su come creare l'esempioAPI, consultathe section called “Tutorial: creazione di un'API REST mediante l'importazione di un esempio”.

Abilita CORS su una risorsa utilizzando l'importazione da API Gateway API

Se si utilizza APIGateway Import API, è possibile configurare CORS il supporto utilizzando un API file Open. È prima necessario definire un metodo `OPTIONS` nella risorsa che restituisce le intestazioni necessarie.

> **Note**
>
> I browser Web prevedono che le intestazioni Access-Control-Allow-Headers e Access-Control-Allow-Origin vengano impostate in ogni metodo che accetta le richieste. API CORS Inoltre, alcuni browser effettuano prima una HTTP richiesta a un `OPTIONS` metodo nella stessa risorsa e poi si aspettano di ricevere le stesse intestazioni.

Metodo di esempio **`Options`**

L'esempio seguente crea un metodo `OPTIONS` per un'integrazione fittizia.

OpenAPI 3.0

```
/users:
```

```
  options:
    summary: CORS support
    description: |
      Enable CORS by returning correct headers
    tags:
    - CORS
    responses:
      200:
        description: Default response for CORS method
        headers:
          Access-Control-Allow-Origin:
            schema:
              type: "string"
          Access-Control-Allow-Methods:
            schema:
              type: "string"
          Access-Control-Allow-Headers:
            schema:
              type: "string"
        content: {}
    x-amazon-apigateway-integration:
      type: mock
      requestTemplates:
        application/json: "{\"statusCode\": 200}"
      passthroughBehavior: "never"
      responses:
        default:
          statusCode: "200"
          responseParameters:
            method.response.header.Access-Control-Allow-Headers: "'Content-Type,X-
Amz-Date,Authorization,X-Api-Key'"
            method.response.header.Access-Control-Allow-Methods: "'*'"
            method.response.header.Access-Control-Allow-Origin: "'*'"
```

OpenAPI 2.0

```
/users:
   options:
      summary: CORS support
      description: |
        Enable CORS by returning correct headers
      consumes:
```

```
        - "application/json"
      produces:
        - "application/json"
      tags:
        - CORS
      x-amazon-apigateway-integration:
        type: mock
        requestTemplates: "{\"statusCode\": 200}"
        passthroughBehavior: "never"
        responses:
          "default":
            statusCode: "200"
            responseParameters:
              method.response.header.Access-Control-Allow-Headers : "'Content-
Type,X-Amz-Date,Authorization,X-Api-Key'"
              method.response.header.Access-Control-Allow-Methods : "'*'"
              method.response.header.Access-Control-Allow-Origin : "'*'"
      responses:
        200:
          description: Default response for CORS method
          headers:
            Access-Control-Allow-Headers:
              type: "string"
            Access-Control-Allow-Methods:
              type: "string"
            Access-Control-Allow-Origin:
              type: "string"
```

Dopo aver configurato il `OPTIONS` metodo per la risorsa, è possibile aggiungere le intestazioni richieste agli altri metodi della stessa risorsa che devono accettare CORS le richieste.

1. Dichiarare Access-Control-Allow-Origin (Controllo accessi - Autorizza origine) e Headers (Intestazioni) nei tipi di risposta.

   OpenAPI 3.0

   ```
   responses:
     200:
       description: Default response for CORS method
       headers:
         Access-Control-Allow-Origin:
           schema:
   ```

```
              type: "string"
          Access-Control-Allow-Methods:
            schema:
              type: "string"
          Access-Control-Allow-Headers:
            schema:
              type: "string"
        content: {}
```

OpenAPI 2.0

```
    responses:
        200:
          description: Default response for CORS method
          headers:
            Access-Control-Allow-Headers:
              type: "string"
            Access-Control-Allow-Methods:
              type: "string"
            Access-Control-Allow-Origin:
              type: "string"
```

2. Nel tag `x-amazon-apigateway-integration` tag configura la mappatura per quelle intestazioni nei valori statici:

   OpenAPI 3.0

```
    responses:
        default:
          statusCode: "200"
          responseParameters:
            method.response.header.Access-Control-Allow-Headers: "'Content-
Type,X-Amz-Date,Authorization,X-Api-Key'"
            method.response.header.Access-Control-Allow-Methods: "'*'"
            method.response.header.Access-Control-Allow-Origin: "'*'"
          responseTemplates:
            application/json: |
              {}
```

   OpenAPI 2.0

```
    responses:
```

```
          "default":
            statusCode: "200"
            responseParameters:
              method.response.header.Access-Control-Allow-Headers : "'Content-
Type,X-Amz-Date,Authorization,X-Api-Key'"
              method.response.header.Access-Control-Allow-Methods : "'*'"
              method.response.header.Access-Control-Allow-Origin : "'*'"
```

Esempio API

L'esempio seguente crea un completo API con un `OPTIONS` metodo e un `GET` metodo con un'`HTTP`integrazione.

OpenAPI 3.0

```
openapi: "3.0.1"
info:
  title: "cors-api"
  description: "cors-api"
  version: "2024-01-16T18:36:01Z"
servers:
- url: "/{basePath}"
  variables:
    basePath:
      default: "/test"
paths:
  /:
    get:
      operationId: "GetPet"
      responses:
        "200":
          description: "200 response"
          headers:
            Access-Control-Allow-Origin:
              schema:
                type: "string"
          content: {}
      x-amazon-apigateway-integration:
        httpMethod: "GET"
        uri: "http://petstore.execute-api.us-east-1.amazonaws.com/petstore/pets"
        responses:
          default:
```

```
            statusCode: "200"
            responseParameters:
              method.response.header.Access-Control-Allow-Origin: "'*'"
        passthroughBehavior: "never"
        type: "http"
    options:
      responses:
        "200":
          description: "200 response"
          headers:
            Access-Control-Allow-Origin:
              schema:
                type: "string"
            Access-Control-Allow-Methods:
              schema:
                type: "string"
            Access-Control-Allow-Headers:
              schema:
                type: "string"
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/Empty"
      x-amazon-apigateway-integration:
        responses:
          default:
            statusCode: "200"
            responseParameters:
              method.response.header.Access-Control-Allow-Methods: "'GET,OPTIONS'"
              method.response.header.Access-Control-Allow-Headers: "'Content-Type,X-
Amz-Date,Authorization,X-Api-Key'"
              method.response.header.Access-Control-Allow-Origin: "'*'"
        requestTemplates:
          application/json: "{\"statusCode\": 200}"
        passthroughBehavior: "never"
        type: "mock"
components:
  schemas:
    Empty:
      type: "object"
```

OpenAPI 2.0

```
swagger: "2.0"
info:
  description: "cors-api"
  version: "2024-01-16T18:36:01Z"
  title: "cors-api"
basePath: "/test"
schemes:
- "https"
paths:
  /:
    get:
      operationId: "GetPet"
      produces:
      - "application/json"
      responses:
        "200":
          description: "200 response"
          headers:
            Access-Control-Allow-Origin:
              type: "string"
      x-amazon-apigateway-integration:
        httpMethod: "GET"
        uri: "http://petstore.execute-api.us-east-1.amazonaws.com/petstore/pets"
        responses:
          default:
            statusCode: "200"
            responseParameters:
              method.response.header.Access-Control-Allow-Origin: "'*'"
        passthroughBehavior: "never"
        type: "http"
    options:
      consumes:
      - "application/json"
      produces:
      - "application/json"
      responses:
        "200":
          description: "200 response"
          schema:
            $ref: "#/definitions/Empty"
          headers:
            Access-Control-Allow-Origin:
```

```
              type: "string"
            Access-Control-Allow-Methods:
              type: "string"
            Access-Control-Allow-Headers:
              type: "string"
      x-amazon-apigateway-integration:
        responses:
          default:
            statusCode: "200"
            responseParameters:
              method.response.header.Access-Control-Allow-Methods: "'GET,OPTIONS'"
              method.response.header.Access-Control-Allow-Headers: "'Content-Type,X-
Amz-Date,Authorization,X-Api-Key'"
              method.response.header.Access-Control-Allow-Origin: "'*'"
        requestTemplates:
          application/json: "{\"statusCode\": 200}"
        passthroughBehavior: "never"
        type: "mock"
definitions:
  Empty:
    type: "object"
```

Test CORS per un API gateway API

Puoi testare la tua CORS configurazione API richiamando la tua API e controllando le CORS intestazioni nella risposta. Il `curl` comando seguente invia una OPTIONS richiesta a un distribuito. API

```
curl -v -X OPTIONS https://{restapi_id}.execute-api.{region}.amazonaws.com/{stage_name}
```

```
< HTTP/1.1 200 OK
< Date: Tue, 19 May 2020 00:55:22 GMT
< Content-Type: application/json
< Content-Length: 0
< Connection: keep-alive
< x-amzn-RequestId: a1b2c3d4-5678-90ab-cdef-abc123
< Access-Control-Allow-Origin: *
< Access-Control-Allow-Headers: Content-Type,Authorization,X-Amz-Date,X-Api-Key,X-Amz-
Security-Token
< x-amz-apigw-id: Abcd=
< Access-Control-Allow-Methods: DELETE,GET,HEAD,OPTIONS,PATCH,POST,PUT
```

Le `Access-Control-Allow-Methods` intestazioni `Access-Control-Allow-OriginAccess-Control-Allow-Headers`, e nella risposta mostrano che i API supporti. CORS Per ulteriori informazioni, consulta CORSper REST APIs in API Gateway.

Tipi di supporti binari per REST APIs in API Gateway

In API Gateway, la API richiesta e la risposta hanno un payload testuale o binario. Un payload di testo è una stringa con `UTF-8` codificaJSON. mentre un payload binario è rappresentato da tutto ciò che non è un payload di testo. Il payload binario può essere, ad esempio, un JPEG file, un file o un GZip file. XML La API configurazione richiesta per supportare i supporti binari dipende dal fatto che vengano API utilizzate integrazioni proxy o non proxy.

AWS Lambda integrazioni proxy

Per gestire i payload binari per le integrazioni AWS Lambda proxy, è necessario codificare in base 64 la risposta della funzione. È inoltre necessario configurare il per. binaryMediaTypesAPI La tua `binaryMediaTypes` configurazione API è un elenco di tipi di contenuto che API tratti come dati binari. I tipi di supporti binari di esempio includono `image/png` o `application/octet-stream`. È possibile utilizzare il carattere jolly (`*`) per includere più tipi di file multimediali. Ad esempio, `*/*` include tutti i tipi di contenuti.

Per il codice di esempio, consulta the section called “Restituisci supporti binari da un'integrazione proxy Lambda in Gateway API”.

Integrazioni non proxy

Per gestire i payload binari per le integrazioni non proxy, aggiungi i tipi di file multimediali all'binaryMediaTypeselenco della risorsa. `RestApi` La tua `binaryMediaTypes` configurazione API è un elenco di tipi di contenuto che API tratti come dati binari. In alternativa, puoi impostare le contentHandlingproprietà sull'integrazione e sulle IntegrationResponserisorse. Il valore `contentHandling` può essere `CONVERT_TO_BINARY`, `CONVERT_TO_TEXT` o non definito.

A seconda del `contentHandling` valore e se l'intestazione della risposta o dell'`Content-Type`intestazione della richiesta in entrata corrisponde a una voce dell'`binaryMediaTypes`elenco, API Gateway può codificare i byte binari non elaborati come stringa con codifica Base64, decodificare una stringa con codifica Base64 riportandola ai relativi byte non elaborati o passare il corpo senza modifiche. `Accept`

È necessario configurare quanto segue per supportare API i payload binari per il sistema in Gateway: API API

- Aggiungete i tipi di file multimediali binari desiderati all'`binaryMediaTypes`elenco della RestApirisorsa. Se questa proprietà e la `contentHandling` proprietà non sono definite, i payload vengono gestiti come UTF -8 stringhe JSON codificate.
- Indirizzare la proprietà `contentHandling` della risorsa Integration.
  - Per convertire il payload della richiesta da una stringa con codifica base64 nel relativo BLOB binario, impostare la proprietà su `CONVERT_TO_BINARY`.
  - Per convertire il payload della richiesta da un BLOB binario a una stringa con codifica base64, impostare la proprietà su `CONVERT_TO_TEXT`.
  - Per passare il payload senza modifiche, lasciare la proprietà indefinita. Per far passare un payload binario senza modifiche, è inoltre necessario assicurarsi che `Content-Type` corrisponda a una delle `binaryMediaTypes` voci e che i comportamenti passthrough siano abilitati per. API
- Imposta la `contentHandling` proprietà della risorsa. IntegrationResponse La `contentHandling` proprietà, l'`Accept`intestazione nelle richieste dei client e la `binaryMediaTypes` combinazione delle tue API determinano il modo in cui API Gateway gestisce le conversioni dei tipi di contenuto. Per informazioni dettagliate, vedi the section called “Conversioni dei tipi di contenuto in Gateway API”.

> **Important**
>
> Quando una richiesta contiene più tipi di file multimediali nell'`Accept`intestazione, API Gateway rispetta solo il primo tipo di supporto. `Accept` Se non riesci a controllare l'ordine dei tipi di `Accept` file multimediali e il tipo di file multimediale dei tuoi contenuti binari non è il primo dell'elenco, aggiungi il primo tipo di `Accept` file multimediale nell'`binaryMediaTypes`elenco. API APIGateway gestisce tutti i tipi di contenuto in questo elenco come binari.
> Ad esempio, per inviare un JPEG file utilizzando un `<img>` elemento in un browser, il browser potrebbe inviare `Accept:image/webp,image/*,*/*;q=0.8` una richiesta. Aggiungendo `image/webp` all'`binaryMediaTypes`elenco, l'endpoint riceve il JPEG file come binario.

Per informazioni dettagliate su come API Gateway gestisce i payload di testo e binari, vedere. Conversioni dei tipi di contenuto in Gateway API

Conversioni dei tipi di contenuto in Gateway API

La combinazione dei tuoi API`binaryMediaTypes`, le intestazioni nelle richieste dei client e la `contentHandling` proprietà di integrazione determinano il modo in cui API Gateway codifica i payload.

La tabella seguente mostra come API Gateway converte il payload della richiesta per configurazioni specifiche dell'`Content-Type`intestazione di una richiesta, l'`binaryMediaTypes`elenco di una RestApirisorsa e il valore della proprietà della risorsa di integrazione. `contentHandling`

Payload della richiesta di metodo	Intestazione **`Content-Type`** della richiesta	**`binaryMediaTypes`**	**`contentHandling`**	Payload della richiesta di integrazione
Dati di testo	Qualsiasi tipo di dato	Undefined	Undefined	UTF8stringa codificata
Dati di testo	Qualsiasi tipo di dato	Undefined	`CONVERT_TO_BINARY`	BLOB binario con codifica Base64
Dati di testo	Qualsiasi tipo di dato	Undefined	`CONVERT_TO_TEXT`	UTF8stringa codificata
Dati di testo	Un tipo di dati di testo	Imposta con tipi di supporto corrispondenti	Undefined	Dati di testo
Dati di testo	Un tipo di dati di testo	Imposta con tipi di supporto corrispondenti	`CONVERT_TO_BINARY`	BLOB binario con codifica Base64
Dati di testo	Un tipo di dati di testo	Imposta con tipi di supporto corrispondenti	`CONVERT_TO_TEXT`	Dati di testo

Payload della richiesta di metodo	Intestazione **`Content-Type`** della richiesta	**`binaryMediaTypes`**	**`contentHandling`**	Payload della richiesta di integrazione
Dati binari	Un tipo di dati binari	Imposta con tipi di supporto corrispondenti	Undefined	Dati binari
Dati binari	Un tipo di dati binari	Imposta con tipi di supporto corrispondenti	`CONVERT_TO_BINARY`	Dati binari
Dati binari	Un tipo di dati binari	Imposta con tipi di supporto corrispondenti	`CONVERT_TO_TEXT`	Stringa con codifica Base64

La tabella seguente mostra come API Gateway converte il payload di risposta per configurazioni specifiche dell'`Accept`intestazione di una richiesta, l'`binaryMediaTypes`elenco di una RestApirisorsa e il valore della proprietà della `contentHandling` risorsa. IntegrationResponse

> **Important**
>
> Quando una richiesta contiene più tipi di file multimediali nell'`Accept`intestazione, API Gateway rispetta solo il primo tipo di supporto. `Accept` Se non riesci a controllare l'ordine dei tipi di `Accept` file multimediali e il tipo di file multimediale dei tuoi contenuti binari non è il primo dell'elenco, aggiungi il primo tipo di `Accept` file multimediale nell'`binaryMediaTypes`elenco. API APIGateway gestisce tutti i tipi di contenuto in questo elenco come binari.
>
> Ad esempio, per inviare un JPEG file utilizzando un `<img>` elemento in un browser, il browser potrebbe inviare `Accept:image/webp,image/*,*/*;q=0.8` una richiesta. Aggiungendo `image/webp` all'`binaryMediaTypes`elenco, l'endpoint riceve il JPEG file come binario.

Payload della risposta di integrazione	Intestazione **`Accept`** della richiesta	**`binaryMediaTypes`**	**`contentHandling`**	Payload della risposta del metodo
Dati di testo o binari	Un tipo di testo	Undefined	Undefined	UTF8stringa codificata
Dati di testo o binari	Un tipo di testo	Undefined	`CONVERT_TO_BINARY`	BLOB con codifica Base64
Dati di testo o binari	Un tipo di testo	Undefined	`CONVERT_TO_TEXT`	UTF8stringa codificata
Dati di testo	Un tipo di testo	Imposta con tipi di supporto corrispondenti	Undefined	Dati di testo
Dati di testo	Un tipo di testo	Imposta con tipi di supporto corrispondenti	`CONVERT_TO_BINARY`	BLOB con codifica Base64
Dati di testo	Un tipo di testo	Imposta con tipi di supporto corrispondenti	`CONVERT_TO_TEXT`	UTF8stringa codificata
Dati di testo	Un tipo binario	Imposta con tipi di supporto corrispondenti	Undefined	BLOB con codifica Base64
Dati di testo	Un tipo binario	Imposta con tipi di supporto corrispondenti	`CONVERT_TO_BINARY`	BLOB con codifica Base64
Dati di testo	Un tipo binario	Imposta con tipi di supporto corrispondenti	`CONVERT_TO_TEXT`	UTF8stringa codificata

Payload della risposta di integrazione	Intestazione **`Accept`** della richiesta	**`binaryMediaTypes`**	**`contentHandling`**	Payload della risposta del metodo
Dati binari	Un tipo di testo	Imposta con tipi di supporto corrispondenti	Undefined	Stringa con codifica Base64
Dati binari	Un tipo di testo	Imposta con tipi di supporto corrispondenti	`CONVERT_TO_BINARY`	Dati binari
Dati binari	Un tipo di testo	Imposta con tipi di supporto corrispondenti	`CONVERT_TO_TEXT`	Stringa con codifica Base64
Dati binari	Un tipo binario	Imposta con tipi di supporto corrispondenti	Undefined	Dati binari
Dati binari	Un tipo binario	Imposta con tipi di supporto corrispondenti	`CONVERT_TO_BINARY`	Dati binari
Dati binari	Un tipo binario	Imposta con tipi di supporto corrispondenti	`CONVERT_TO_TEXT`	Stringa con codifica Base64

Quando converte un payload di testo in un blob binario, API Gateway presuppone che i dati di testo siano una stringa codificata in base64 e restituisce i dati binari come blob decodificati in base64. Se la conversione fallisce, restituisce una risposta che indica un errore di configurazione. `500` API Non viene fornito un modello di mappatura per tale conversione, sebbene sia necessario abilitare i comportamenti passthrough su. API

Quando si converte un payload binario in una stringa di testo, API Gateway applica sempre una codifica base64 sui dati binari. Per tale payload puoi definire un modello di mappatura, ma puoi accedere alla stringa con codifica base64 nel modello di mappatura solo attraverso `$input.body`, come viene mostrato nel seguente estratto di un modello di mappatura di esempio.

```
{
    "data": "$input.body"
}
```

Per far passare il payload binario senza modifiche, è necessario abilitare i comportamenti passthrough su. API

Abilitazione del supporto binario tramite la console Gateway API

La sezione spiega come abilitare il supporto binario utilizzando la console API Gateway. Ad esempio, utilizziamo un API sistema integrato con Amazon S3. Ci focalizziamo sulle attività per impostare i tipi di supporti ammessi e per specificare come dovrebbe essere gestito il payload. Per informazioni dettagliate su come creare un API sistema integrato con Amazon S3, consulta. Tutorial: creare un REST API proxy come Amazon S3

Per abilitare il supporto binario utilizzando la console API Gateway

1. Imposta i tipi di supporti binari perAPI:
   a. Creane uno nuovo API o scegline uno esistenteAPI. Per questo esempio, chiamiamo API`FileMan`.
   b. Nella sezione selezionata API nel pannello di navigazione principale, scegli APIimpostazioni.
   c. Nel riquadro delle APIimpostazioni, scegli Gestisci i tipi di file multimediali nella sezione Tipi di file multimediali binari.
   d. Scegli Aggiungi tipo di supporto binario.
   e. Immetti il tipo di supporto richiesto, ad esempio **`image/png`**, nel campo di input. Se necessario, ripeti questa fase per aggiungere altri tipi di supporto. Per supportare tutti i tipi di file multimediali binari, specifica `*/*`.
   f. Seleziona Salvataggio delle modifiche.
2. Imposta come vengono gestiti i payload dei messaggi per il API metodo:
   a. Crea una nuova risorsa o scegli una risorsa esistente in. API Ad esempio, noi utilizziamo la risorsa `/{folder}/{item}`.
   b. Crea un nuovo metodo o scegli un metodo esistente della risorsa. Come esempio, utilizziamo il metodo `GET /{folder}/{item}` integrato nell'azione `Object GET` in Amazon S3.
   c. Per Gestione contenuti scegli un'opzione.

Scegliere Passthrough se non si vuole convertire il corpo quando il client e il back-end accettano lo stesso formato binario. Scegliete Converti in testo per convertire il corpo binario in una stringa con codifica Base64 quando, ad esempio, il backend richiede che il payload di una richiesta binaria venga passato come proprietà. JSON Scegli quindi Converti in binario quando il client invia una stringa con codifica base64 e il back-end richiede il formato binario originale o quando l'endpoint restituisce una stringa con codifica base64 e il client accetta solo l'output binario.

d. Per Richiesta corpo passthrough scegli Quando non ci sono modelli definiti (consigliato).

   Puoi anche scegliere Mai. Ciò significa che API rifiuterà i dati con tipi di contenuto che non dispongono di un modello di mappatura.

e. Mantieni l'intestazione `Accept` della richiesta in entrata nella richiesta di integrazione. Procedi in questo modo se hai impostato `contentHandling` su `passthrough` e vuoi sovrascrivere questa impostazione al runtime.

HTTP headers (2) 1

Name	Mapped from	Caching
Accept	method.request.header.Accept	Inactive
Content-Type	method.request.header.Content-Type	Inactive

f. Per la conversione in testo, definisci un modello di mappatura per mettere i dati binari con codifica base64 nel formato richiesto.

   Un esempio di modello di mappatura per la conversione in testo è il seguente:

```
{
  "operation": "thumbnail",
  "base64Image": "$input.body"
}
```

   Il formato di questo modello di mappatura dipende dai requisiti dell'endpoint dell'input.

g. Selezionare Salva.

Abilitazione del supporto binario tramite il Gateway API REST API

Le seguenti attività mostrano come abilitare il supporto binario utilizzando le REST API chiamate API Gateway.

Argomenti

- Aggiungere e aggiornare i tipi di file multimediali binari supportati a un API
- Configurazione delle conversioni dei payload delle richieste
- Configurazione delle conversioni dei payload delle risposte
- Conversione dei dati binari in dati di testo
- Conversione dei dati di testo in un payload binario
- Passaggio attraverso un payload binario

Aggiungere e aggiornare i tipi di file multimediali binari supportati a un API

Per consentire a API Gateway di supportare un nuovo tipo di supporto binario, è necessario aggiungere il tipo di supporto binario all'`binaryMediaTypes`elenco della `RestApi` risorsa. Ad esempio, per fare in modo che API Gateway gestisca JPEG le immagini, inviate una `PATCH` richiesta alla `RestApi` risorsa:

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/image~1jpeg"
  }
 ]
}
```

La specifica del MIME tipo `image/jpeg` che fa parte del valore della `path` proprietà viene espulsa come`image~1jpeg`.

Per aggiornare i tipi di supporto binario utilizzati, sostituisci o elimina i tipi di supporto dall'elenco `binaryMediaTypes` della risorsa `RestApi`. Ad esempio, per modificare il supporto binario dai JPEG file ai byte non elaborati, invia una `PATCH` richiesta alla `RestApi` risorsa nel modo seguente:

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [{
    "op" : "replace",
    "path" : "/binaryMediaTypes/image~1jpeg",
    "value" : "application/octet-stream"
  },
  {
    "op" : "remove",
    "path" : "/binaryMediaTypes/image~1jpeg"
  }]
}
```

Configurazione delle conversioni dei payload delle richieste

Se l'endpoint richiede un input binario, imposta la proprietà `contentHandling` della risorsa `Integration` su `CONVERT_TO_BINARY`. A tale scopo, invia una richiesta `PATCH`, come segue:

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/integration

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "/contentHandling",
    "value" : "CONVERT_TO_BINARY"
  }]
}
```

Configurazione delle conversioni dei payload delle risposte

Se il client accetta il risultato come BLOB binario invece del payload con codifica base64 restituito dall'endpoint, imposta la proprietà `contentHandling` della risorsa `IntegrationResponse` su `CONVERT_TO_BINARY`. A questo scopo, inviare una richiesta `PATCH`, come segue:

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/integration/
responses/<status_code>

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "/contentHandling",
    "value" : "CONVERT_TO_BINARY"
  }]
}
```

Conversione dei dati binari in dati di testo

Per inviare dati binari come JSON proprietà dell'input a AWS Lambda o Kinesis tramite API Gateway, procedi come segue:

1. Abilita il supporto per il payload binario di API aggiungendo il nuovo tipo di supporto binario API di `application/octet-stream` `binaryMediaTypes` all'elenco.

   ```
   PATCH /restapis/<restapi_id>

   {
     "patchOperations" : [ {
       "op" : "add",
       "path" : "/binaryMediaTypes/application~1octet-stream"
   ```

```
  }
 ]
}
```

2. `CONVERT_TO_TEXT`Imposta la `contentHandling` proprietà della `Integration` risorsa e fornisci un modello di mappatura per assegnare la stringa codificata in base64 dei dati binari a una proprietà. JSON Nell'esempio seguente, la JSON proprietà è e contiene la stringa con codifica base64. body `$input.body`

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/
integration

{
  "patchOperations" : [
    {
      "op" : "replace",
      "path" : "/contentHandling",
      "value" : "CONVERT_TO_TEXT"
    },
    {
      "op" : "add",
      "path" : "/requestTemplates/application~1octet-stream",
      "value" : "{\"body\": \"$input.body\"}"
    }
  ]
}
```

Conversione dei dati di testo in un payload binario

Supponiamo che una funzione Lambda restituisca un file di immagine come stringa con codifica base64. Per passare questo output binario al client tramite API Gateway, effettuate le seguenti operazioni:

1. Aggiorna API l'`binaryMediaTypes`elenco aggiungendo il tipo di supporto binario di`application/octet-stream`, se non è già presente nell'elenco.

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
```

```
    "path" : "/binaryMediaTypes/application~1octet-stream",
  }]
}
```

2. Imposta la proprietà `contentHandling` della risorsa `Integration` su `CONVERT_TO_BINARY`. Non definire un modello di mappatura. Se non si definisce un modello di mappatura, API Gateway richiama il modello passthrough per restituire il blob binario decodificato in base64 come file di immagine al client.

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/
integration/responses/<status_code>

{
  "patchOperations" : [
    {
      "op" : "replace",
      "path" : "/contentHandling",
      "value" : "CONVERT_TO_BINARY"
    }
  ]
}
```

Passaggio attraverso un payload binario

Per archiviare un'immagine in un bucket Amazon S3 utilizzando API Gateway, procedi come segue:

1. Aggiorna API l'`binaryMediaTypes`elenco aggiungendo il tipo di supporto binario di`application/octet-stream`, se non è già presente nell'elenco.

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/application~1octet-stream"
  }
 ]
}
```

2. Imposta la proprietà `contentHandling` della risorsa `Integration` su `CONVERT_TO_BINARY`. Imposta `WHEN_NO_MATCH` come il valore della proprietà `passthroughBehavior` senza definire un modello di mappatura. Ciò consente a API Gateway di richiamare il modello passthrough.

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/
integration

{
  "patchOperations" : [
    {
      "op" : "replace",
      "path" : "/contentHandling",
      "value" : "CONVERT_TO_BINARY"
    },
    {
      "op" : "replace",
      "path" : "/passthroughBehaviors",
      "value" : "WHEN_NO_MATCH"
    }
  ]
}
```

Importa ed esporta le codifiche dei contenuti per Gateway API

Per importare l'`binaryMediaTypes`elenco su un file RestApi, utilizzate la seguente estensione API Gateway nel file API di API definizione Open. L'estensione viene utilizzata anche per esportare le API impostazioni.

- x-amazon-apigateway-binaryproprietà -media-types

Per importare ed esportare il valore della `contentHandling` proprietà su una `IntegrationResponse` risorsa `Integration` or, utilizzate le seguenti estensioni API Gateway per le API definizioni Open:

- x-amazon-apigateway-integration oggetto
- x-amazon-apigateway-integrationoggetto.response

Restituisci supporti binari da un'integrazione proxy Lambda in Gateway API

Per restituire un supporto binario da un'integrazione proxy AWS Lambda, codificare in base64 la risposta dalla funzione Lambda. È inoltre necessario configurare i tipi API di file multimediali binari. Il limite della dimensione del payload è 10 MB.

> **Note**
>
> Per utilizzare un browser Web per richiamare un'integrazione API con questo esempio, imposta i tipi API di file multimediali binari su`*/*`. APIGateway utilizza la prima `Accept` intestazione dei client per determinare se una risposta deve restituire un supporto binario. Per restituire file multimediali binari quando non puoi controllare l'ordine dei valori di `Accept` intestazione, ad esempio le richieste provenienti da un browser, imposta i tuoi tipi API di file multimediali binari su `*/*` (per tutti i tipi di contenuto).

Il seguente esempio di funzione Lambda può restituire ai client un'immagine binaria da Amazon S3 o del testo. La risposta della funzione include un'intestazione `Content-Type` per indicare al client il tipo di dati che restituisce. La funzione imposta in modo condizionale la proprietà `isBase64Encoded` nella risposta, a seconda del tipo di dati che restituisce.

Node.js

```
import { S3Client, GetObjectCommand } from "@aws-sdk/client-s3"

const client = new S3Client({region: 'us-east-2'});

export const handler = async (event) => {

  var randomint = function(max) {
    return Math.floor(Math.random() * max);
  }
  var number = randomint(2);
  if (number == 1){
    const input = {
      "Bucket" : "bucket-name",
      "Key" : "image.png"
      }
    try {
      const command = new GetObjectCommand(input)
      const response = await client.send(command);
```

```
      var str = await response.Body.transformToByteArray();
    } catch (err) {
      console.error(err);
    }
    const base64body = Buffer.from(str).toString('base64');
    return {
      'headers': { "Content-Type": "image/png" },
      'statusCode': 200,
      'body': base64body,
      'isBase64Encoded': true
      }
    } else {
        return {
        'headers': { "Content-Type": "text/html" },
        'statusCode': 200,
        'body': "<h1>This is text</h1>",
        }
    }
}
```

Python

```
import base64
import boto3
import json
import random

s3 = boto3.client('s3')

def lambda_handler(event, context):
    number = random.randint(0,1)
    if number == 1:
        response = s3.get_object(
            Bucket='bucket-name',
            Key='image.png',
        )
        image = response['Body'].read()
        return {
            'headers': { "Content-Type": "image/png" },
            'statusCode': 200,
            'body': base64.b64encode(image).decode('utf-8'),
            'isBase64Encoded': True
        }
```

```
    else:
        return {
            'headers': { "Content-type": "text/html" },
            'statusCode': 200,
            'body': "<h1>This is text</h1>",
        }
```

Per ulteriori informazioni sui tipi di supporti binari, consulta Tipi di supporti binari per REST APIs in API Gateway.

Accedi ai file binari in Amazon S3 tramite un gateway API API

Gli esempi seguenti mostrano il API file Open utilizzato per accedere alle immagini in Amazon S3, come scaricare un'immagine da Amazon S3 e come caricare un'immagine su Amazon S3.

Argomenti

- Apri API il file di un campione API per accedere alle immagini in Amazon S3
- Download di un'immagine da Amazon S3
- Caricamento di un'immagine su Amazon S3

Apri API il file di un campione API per accedere alle immagini in Amazon S3

Il seguente API file aperto mostra un esempio API che illustra il download di un file di immagine da Amazon S3 e il caricamento di un file di immagine su Amazon S3. Questo illustra API i `PUT /s3?key={file-name}` metodi `GET /s3?key={file-name}` e i metodi per scaricare e caricare un file di immagine specificato. Il `GET` metodo restituisce il file di immagine come stringa codificata in base64 come parte di un JSON output, seguendo il modello di mappatura fornito, in una risposta di 200 OK. Il metodo PUT prende un BLOB binario non elaborato come input e restituisce una risposta 200 OK con un payload vuoto.

OpenAPI 3.0

```
{
   "openapi": "3.0.0",
   "info": {
      "version": "2016-10-21T17:26:28Z",
      "title": "ApiName"
   },
   "paths": {
```

```
      "/s3": {
         "get": {
            "parameters": [
               {
                  "name": "key",
                  "in": "query",
                  "required": false,
                  "schema": {
                     "type": "string"
                  }
               }
            ],
            "responses": {
               "200": {
                  "description": "200 response",
                  "content": {
                     "application/json": {
                        "schema": {
                           "$ref": "#/components/schemas/Empty"
                        }
                     }
                  }
               },
               "500": {
                  "description": "500 response"
               }
            },
            "x-amazon-apigateway-integration": {
               "credentials": "arn:aws:iam::123456789012:role/binarySupportRole",
               "responses": {
                  "default": {
                     "statusCode": "500"
                  },
                  "2\\d{2}": {
                     "statusCode": "200"
                  }
               },
               "requestParameters": {
                  "integration.request.path.key": "method.request.querystring.key"
               },
               "uri": "arn:aws:apigateway:us-west-2:s3:path/{key}",
               "passthroughBehavior": "when_no_match",
               "httpMethod": "GET",
               "type": "aws"
```

```
            }
         },
         "put": {
            "parameters": [
               {
                  "name": "key",
                  "in": "query",
                  "required": false,
                  "schema": {
                     "type": "string"
                  }
               }
            ],
            "responses": {
               "200": {
                  "description": "200 response",
                  "content": {
                     "application/json": {
                        "schema": {
                           "$ref": "#/components/schemas/Empty"
                        }
                     },
                     "application/octet-stream": {
                        "schema": {
                           "$ref": "#/components/schemas/Empty"
                        }
                     }
                  }
               },
               "500": {
                  "description": "500 response"
               }
            },
            "x-amazon-apigateway-integration": {
               "credentials": "arn:aws:iam::123456789012:role/binarySupportRole",
               "responses": {
                  "default": {
                     "statusCode": "500"
                  },
                  "2\\d{2}": {
                     "statusCode": "200"
                  }
               },
               "requestParameters": {
```

```
                  "integration.request.path.key": "method.request.querystring.key"
               },
               "uri": "arn:aws:apigateway:us-west-2:s3:path/{key}",
               "passthroughBehavior": "when_no_match",
               "httpMethod": "PUT",
               "type": "aws",
               "contentHandling": "CONVERT_TO_BINARY"
            }
         }
      }
   },
   "x-amazon-apigateway-binary-media-types": [
      "application/octet-stream",
      "image/jpeg"
   ],
   "servers": [
      {
         "url": "https://abcdefghi.execute-api.us-east-1.amazonaws.com/{basePath}",
         "variables": {
            "basePath": {
              "default": "/v1"
            }
         }
      }
   ],
   "components": {
      "schemas": {
         "Empty": {
            "type": "object",
            "title": "Empty Schema"
         }
      }
   }
}
```

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2016-10-21T17:26:28Z",
    "title": "ApiName"
  },
```

```
  "host": "abcdefghi.execute-api.us-east-1.amazonaws.com",
  "basePath": "/v1",
  "schemes": [
    "https"
  ],
  "paths": {
    "/s3": {
      "get": {
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "key",
            "in": "query",
            "required": false,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          },
          "500": {
            "description": "500 response"
          }
        },
        "x-amazon-apigateway-integration": {
          "credentials": "arn:aws:iam::123456789012:role/binarySupportRole",
          "responses": {
            "default": {
              "statusCode": "500"
            },
            "2\\d{2}": {
              "statusCode": "200"            }
          },
          "requestParameters": {
            "integration.request.path.key": "method.request.querystring.key"
          },
          "uri": "arn:aws:apigateway:us-west-2:s3:path/{key}",
          "passthroughBehavior": "when_no_match",
```

```
          "httpMethod": "GET",
          "type": "aws"
        }
      },
      "put": {
        "produces": [
          "application/json", "application/octet-stream"
        ],
        "parameters": [
          {
            "name": "key",
            "in": "query",
            "required": false,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          },
          "500": {
            "description": "500 response"
          }
        },
        "x-amazon-apigateway-integration": {
          "credentials": "arn:aws:iam::123456789012:role/binarySupportRole",
          "responses": {
            "default": {
              "statusCode": "500"
            },
            "2\\d{2}": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.path.key": "method.request.querystring.key"
          },
          "uri": "arn:aws:apigateway:us-west-2:s3:path/{key}",
          "passthroughBehavior": "when_no_match",
          "httpMethod": "PUT",
          "type": "aws",
```

```
          "contentHandling" : "CONVERT_TO_BINARY"
        }
      }
    }
  },
  "x-amazon-apigateway-binary-media-types" : ["application/octet-stream", "image/
jpeg"],
  "definitions": {
    "Empty": {
      "type": "object",
      "title": "Empty Schema"
    }
  }
}
```

Download di un'immagine da Amazon S3

Per eseguire il download di un file immagine (`image.jpg`) come BLOB binario da Amazon S3:

```
GET /v1/s3?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/json
Accept: application/octet-stream
```

La risposta andata a buon fine avrebbe questa struttura:

```
200 OK HTTP/1.1

[raw bytes]
```

I byte non elaborati vengono restituiti perché l'`Accept`intestazione è impostata su un tipo di supporto binario di e il supporto binario è abilitato per. `application/octet-stream` API

In alternativa, per scaricare un file di immagine (`image.jpg`) come stringa codificata in base64 (formattata come proprietà) JSON da Amazon S3, aggiungi un modello di risposta alla risposta di integrazione 200, come mostrato nel seguente blocco Open definition in grassetto: API

```
        "x-amazon-apigateway-integration": {
          "credentials": "arn:aws:iam::123456789012:role/binarySupportRole",
          "responses": {
            "default": {
```

```
              "statusCode": "500"
            },
            "2\\d{2}": {
              "statusCode": "200",
              "responseTemplates": {
                "application/json": "{\n   \"image\": \"$input.body\"\n}"
              }
            }
          },
```

Di seguito è riportato l'aspetto della richiesta di download di un file di immagine:

```
GET /v1/s3?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/json
Accept: application/json
```

L'aspetto di una risposta andata a buon fine è il seguente:

```
200 OK HTTP/1.1

{
  "image": "W3JhdyBieXRlc10="
}
```

Caricamento di un'immagine su Amazon S3

Per caricare un file immagine (`image.jpg`) come BLOB binario su Amazon S3:

```
PUT /v1/s3?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/octet-stream
Accept: application/json

[raw bytes]
```

L'aspetto di una risposta andata a buon fine è il seguente:

```
200 OK HTTP/1.1
```

Per caricare un file immagine (`image.jpg`) come stringa con codifica base64 su Amazon S3:

```
PUT /v1/s3?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/json
Accept: application/json

W3JhdyBieXRlc10=
```

Il payload di input deve essere una stringa con codifica base64, poiché il valore dell'intestazione `Content-Type` è impostato su `application/json`. L'aspetto di una risposta andata a buon fine è il seguente:

```
200 OK HTTP/1.1
```

Accedi ai file binari in Lambda utilizzando un gateway API API

Il seguente API esempio Open mostra come accedere a un file binario AWS Lambda tramite un Gateway. API API Questo API illustra i metodi `GET /lambda?key={file-name}` e i `PUT /lambda?key={file-name}` metodi per scaricare e caricare un file di immagine specificato. Il `GET` metodo restituisce il file di immagine come stringa codificata in base64 come parte di un JSON output, seguendo il modello di mappatura fornito, in una risposta di 200 OK. Il metodo PUT prende un BLOB binario non elaborato come input e restituisce una risposta 200 OK con un payload vuoto.

Crei la funzione Lambda che API chiami e questa deve restituire una stringa con codifica base64 con l'intestazione di. `Content-Type application/json`

Argomenti

- Apri il API file di un campione API per accedere alle immagini in Lambda
- Download di un'immagine da Lambda
- Caricamento di un'immagine su Lambda

Apri il API file di un campione API per accedere alle immagini in Lambda

Il seguente API file aperto mostra un esempio API che illustra il download di un file di immagine da Lambda e il caricamento di un file di immagine in Lambda.

OpenAPI 3.0

```
{
   "openapi": "3.0.0",
```

```
   "info": {
      "version": "2016-10-21T17:26:28Z",
      "title": "ApiName"
   },
   "paths": {
      "/lambda": {
         "get": {
            "parameters": [
               {
                  "name": "key",
                  "in": "query",
                  "required": false,
                  "schema": {
                     "type": "string"
                  }
               }
            ],
            "responses": {
               "200": {
                  "description": "200 response",
                  "content": {
                     "application/json": {
                        "schema": {
                           "$ref": "#/components/schemas/Empty"
                        }
                     }
                  }
               },
               "500": {
                  "description": "500 response"
               }
            },
            "x-amazon-apigateway-integration": {
               "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:123456789012:function:image/invocations",
               "type": "AWS",
               "credentials": "arn:aws:iam::123456789012:role/Lambda",
               "httpMethod": "POST",
               "requestTemplates": {
                  "application/json": "{\n   \"imageKey\":
 \"$input.params('key')\"\n}"
               },
               "responses": {
                  "default": {
```

```
                    "statusCode": "500"
                },
                "2\\d{2}": {
                    "statusCode": "200",
                    "responseTemplates": {
                        "application/json": "{\n   \"image\": \"$input.body\"\n}"
                    }
                }
            }
        }
    },
    "put": {
        "parameters": [
            {
                "name": "key",
                "in": "query",
                "required": false,
                "schema": {
                    "type": "string"
                }
            }
        ],
        "responses": {
            "200": {
                "description": "200 response",
                "content": {
                    "application/json": {
                        "schema": {
                            "$ref": "#/components/schemas/Empty"
                        }
                    },
                    "application/octet-stream": {
                        "schema": {
                            "$ref": "#/components/schemas/Empty"
                        }
                    }
                }
            },
            "500": {
                "description": "500 response"
            }
        },
        "x-amazon-apigateway-integration": {
```

```
                "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:123456789012:function:image/invocations",
                "type": "AWS",
                "credentials": "arn:aws:iam::123456789012:role/Lambda",
                "httpMethod": "POST",
                "contentHandling": "CONVERT_TO_TEXT",
                "requestTemplates": {
                   "application/json": "{\n   \"imageKey\": \"$input.params('key')\",
 \"image\": \"$input.body\"\n}"
                },
                "responses": {
                   "default": {
                      "statusCode": "500"
                   },
                   "2\\d{2}": {
                      "statusCode": "200"
                   }
                }
             }
          }
       }
    },
    "x-amazon-apigateway-binary-media-types": [
       "application/octet-stream",
       "image/jpeg"
    ],
    "servers": [
       {
          "url": "https://abcdefghi.execute-api.us-east-1.amazonaws.com/{basePath}",
          "variables": {
             "basePath": {
               "default": "/v1"
             }
          }
       }
    ],
    "components": {
       "schemas": {
          "Empty": {
             "type": "object",
             "title": "Empty Schema"
          }
       }
    }
```

```
}
```

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2016-10-21T17:26:28Z",
    "title": "ApiName"
  },
  "host": "abcdefghi.execute-api.us-east-1.amazonaws.com",
  "basePath": "/v1",
  "schemes": [
    "https"
  ],
  "paths": {
    "/lambda": {
      "get": {
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "key",
            "in": "query",
            "required": false,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          },
          "500": {
            "description": "500 response"
          }
        },
        "x-amazon-apigateway-integration": {
          "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789012:function:image/invocations",
```

```
          "type": "AWS",
          "credentials": "arn:aws:iam::123456789012:role/Lambda",
          "httpMethod": "POST",
          "requestTemplates": {
            "application/json": "{\n   \"imageKey\": \"$input.params('key')\"\n}"
          },
          "responses": {
            "default": {
              "statusCode": "500"
            },
            "2\\d{2}": {
              "statusCode": "200",
              "responseTemplates": {
                "application/json": "{\n   \"image\": \"$input.body\"\n}"
              }
            }
          }
        }
      },
      "put": {
        "produces": [
          "application/json", "application/octet-stream"
        ],
        "parameters": [
          {
            "name": "key",
            "in": "query",
            "required": false,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          },
          "500": {
            "description": "500 response"
          }
        },
        "x-amazon-apigateway-integration": {
```

```
          "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789012:function:image/invocations",
          "type": "AWS",
          "credentials": "arn:aws:iam::123456789012:role/Lambda",
          "httpMethod": "POST",
          "contentHandling" : "CONVERT_TO_TEXT",
          "requestTemplates": {
            "application/json": "{\n   \"imageKey\": \"$input.params('key')\",
 \"image\": \"$input.body\"\n}"
          },
          "responses": {
            "default": {
              "statusCode": "500"
            },
            "2\\d{2}": {
              "statusCode": "200"
            }
          }
        }
      }
    }
  },
  "x-amazon-apigateway-binary-media-types" : ["application/octet-stream", "image/
jpeg"],
  "definitions": {
    "Empty": {
      "type": "object",
      "title": "Empty Schema"
    }
  }
}
```

Download di un'immagine da Lambda

Per eseguire il download di un file immagine (`image.jpg`) come BLOB binario da Lambda:

```
GET /v1/lambda?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/json
Accept: application/octet-stream
```

L'aspetto di una risposta andata a buon fine è il seguente:

```
200 OK HTTP/1.1

[raw bytes]
```

Per scaricare un file di immagine (`image.jpg`) come stringa con codifica Base64 (formattata come proprietà) da Lambda: JSON

```
GET /v1/lambda?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/json
Accept: application/json
```

L'aspetto di una risposta andata a buon fine è il seguente:

```
200 OK HTTP/1.1

{
  "image": "W3JhdyBieXRlc10="
}
```

Caricamento di un'immagine su Lambda

Per caricare un file immagine (`image.jpg`) come BLOB binario su Lambda:

```
PUT /v1/lambda?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/octet-stream
Accept: application/json

[raw bytes]
```

L'aspetto di una risposta andata a buon fine è il seguente:

```
200 OK
```

Per caricare un file immagine (`image.jpg`) come stringa con codifica base64 su Lambda:

```
PUT /v1/lambda?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
```

```
Content-Type: application/json
Accept: application/json

W3JhdyBieXRlc10=
```

L'aspetto di una risposta andata a buon fine è il seguente:

```
200 OK
```

Richiama le API REST in API Gateway

Per chiamare un'API distribuita, i client inviano richieste all'URL del servizio del componente API Gateway per l'esecuzione dell'API, ovvero `execute-api`.

L'URL di base per le API REST si presenta nel formato seguente:

```
https://api-id.execute-api.region.amazonaws.com/stage/
```

dove *`api-id`* è l'identificatore dell'API, *`region è la AWS regione`* e *`stage è il nome di fase`* della distribuzione dell'API.

> **Important**
>
> Prima di invocare un'API, devi distribuirla in API Gateway. Per istruzioni sulla distribuzione di un'API, consulta. Implementazione REST APIs in Gateway API

Argomenti

- Ottenere l'URL di richiamo di un'API
- Invocazione di un'API
- Utilizzo della console API Gateway per il test di un metodo API REST
- Utilizzo di un SDK Java generato da API Gateway per un'API REST
- Utilizzo di un SDK Android generato da API Gateway per un'API REST
- Usa un file JavaScript SDK generato da API Gateway per un REST API
- Usa un Ruby SDK generato da API Gateway per un REST API

- Uso dell'SDK iOS generato da API Gateway per un'API REST in Objective-C o Swift

Ottenere l'URL di richiamo di un'API

È possibile utilizzare la console AWS CLI, la o una definizione OpenAPI esportata per ottenere l'URL di richiamo di un'API.

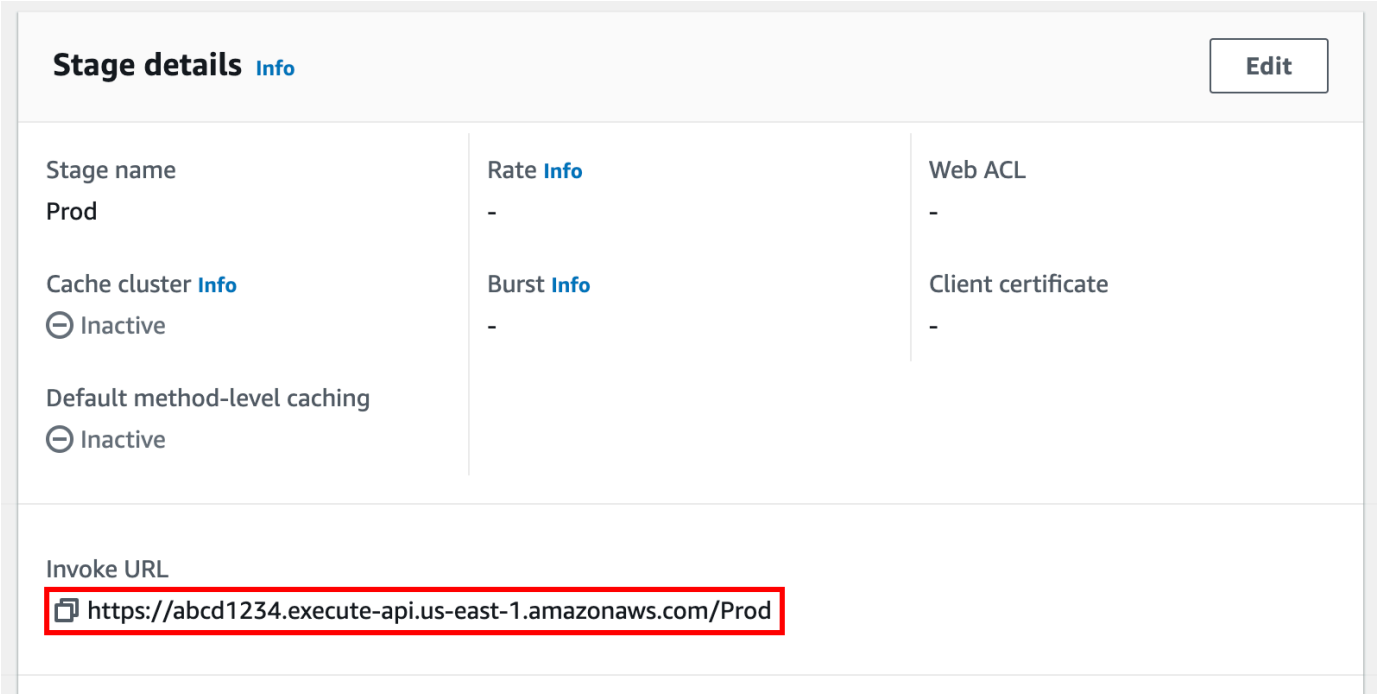
Ottenere l'URL di richiamo di un'API utilizzando la console

La procedura seguente mostra come ottenere l'URL di richiamo di un'API nella console dell'API REST.

Per ottenere l'URL di invocazione di un'API utilizzando la console API REST

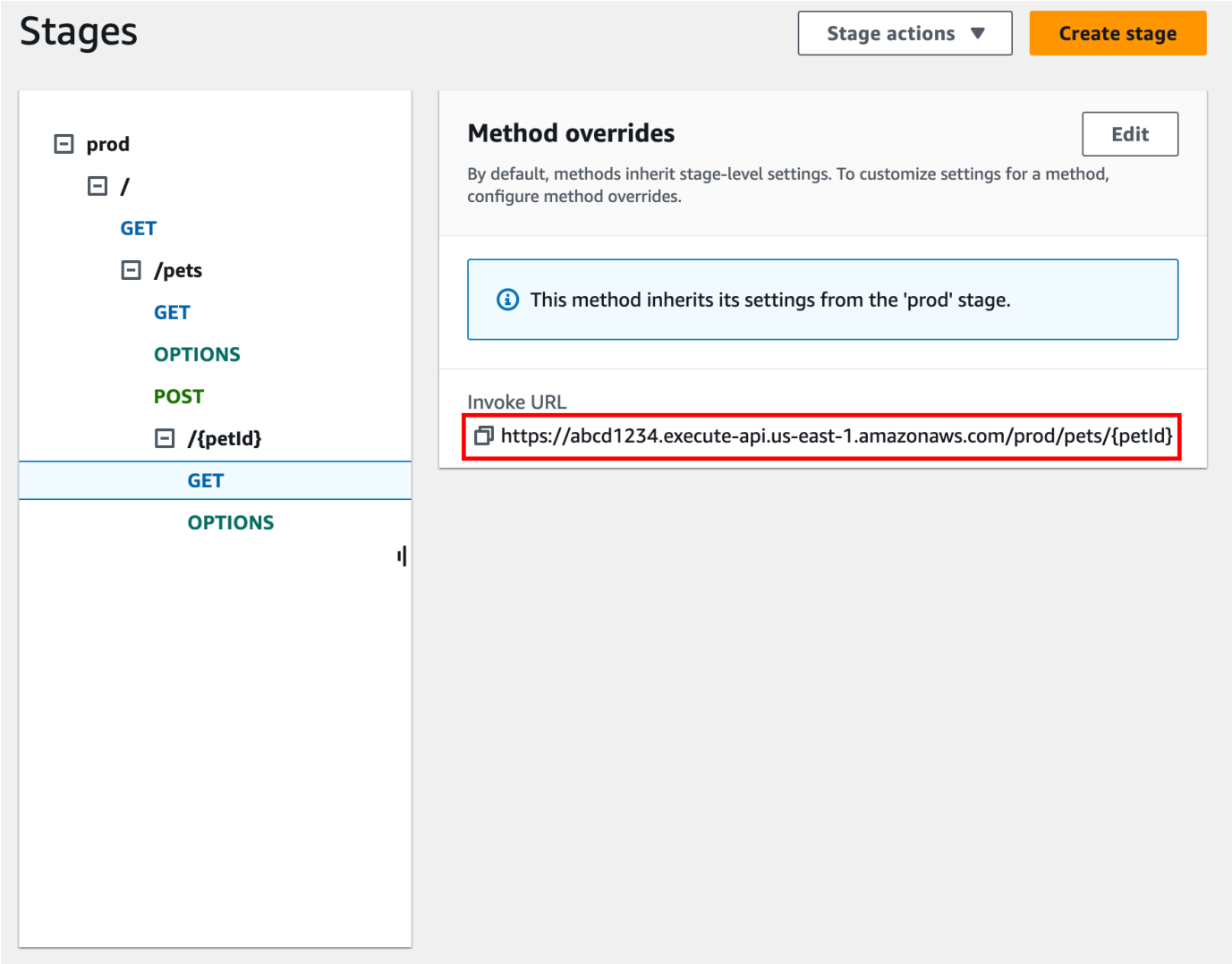
1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli un'API distribuita.
3. Nel riquadro di navigazione principale, seleziona Log.
4. In Dettagli fase, scegli l'icona Copia per copiare l'URL di richiamo dell'API.

   Questo URL è per la risorsa principale della tua API.



5. Per ottenere l'URL di richiamo di un'API per un'altra risorsa dell'API, espandi lo stage nel riquadro di navigazione secondario, quindi scegli un metodo.

6. Scegli l'icona di copia per copiare l'URL di richiamo a livello di risorsa dell'API.



Ottenere l'URL di richiamo di un'API utilizzando il AWS CLI

La procedura seguente mostra come ottenere l'URL di richiamo di un'API utilizzando. AWS CLI

Per ottenere l'URL di richiamo di un'API utilizzando il AWS CLI

1. Utilizzare il comando seguente per ottenere il`rest-api-id`. Questo comando restituisce tutti `rest-api-id` i valori nella tua regione. Per ulteriori informazioni, vedere get-rest-apis.

```
aws apigateway get-rest-apis
```

2. Sostituisci l'esempio `rest-api-id` con il tuo`rest-api-id`, sostituisci l'esempio *`{stage-name}`* con il tuo *`{stage-name}`* e sostituisci *`{region} con la tua Regione`*.

```
https://{restapi_id}.execute-api.{region}.amazonaws.com/{stage_name}/
```

Ottenere l'URL di richiamo di un'API utilizzando il file di definizione OpenAPI esportato dell'API

È inoltre possibile creare l'URL principale combinando `basePath` i campi `host` e di un file di definizione OpenAPI esportato dell'API. Per istruzioni su come esportare la tua API, consulta. the section called “Esporta un REST API”

Invocazione di un'API

Puoi chiamare l'API implementata utilizzando il browser, curl o altre applicazioni, come Postman.

Inoltre, puoi utilizzare la console API Gateway per testare una chiamata API. Test utilizza la `TestInvoke` funzionalità di API Gateway, che consente il test delle API prima che l'API venga distribuita. Per ulteriori informazioni, consulta the section called “Utilizzo della console per il test di un metodo API REST.”.

> **Note**
>
> I valori dei parametri delle stringe di query di un URL di chiamata non possono includere `%%`.

Richiamo di un'API utilizzando un browser Web

Se la tua API consente l'accesso anonimo, puoi utilizzare qualsiasi browser Web per richiamare qualsiasi metodo. `GET` Inserisci l'URL di invocazione completo nella barra degli indirizzi del browser.

Per altri metodi o chiamate che richiedono l'autenticazione, è necessario specificare un payload o firmare le richieste. Puoi gestirle in uno script dietro una pagina HTML o in un'applicazione client utilizzando uno degli SDK. AWS

Invocare un'API usando curl

Puoi usare uno strumento come curl nel tuo terminale per chiamare la tua API. Il seguente comando curl di esempio richiama il metodo GET sulla `getUsers` risorsa dello `prod` stadio di un'API.

Linux or Macintosh

```
curl -X GET 'https://b123abcde4.execute-api.us-west-2.amazonaws.com/prod/getUsers'
```

Windows

```
curl -X GET "https://b123abcde4.execute-api.us-west-2.amazonaws.com/prod/getUsers"
```

Utilizzo della console API Gateway per il test di un metodo API REST

Utilizzo della console API Gateway per il test di un metodo API REST.

Argomenti

- Prerequisiti
- Test di un metodo tramite la console API Gateway

Prerequisiti

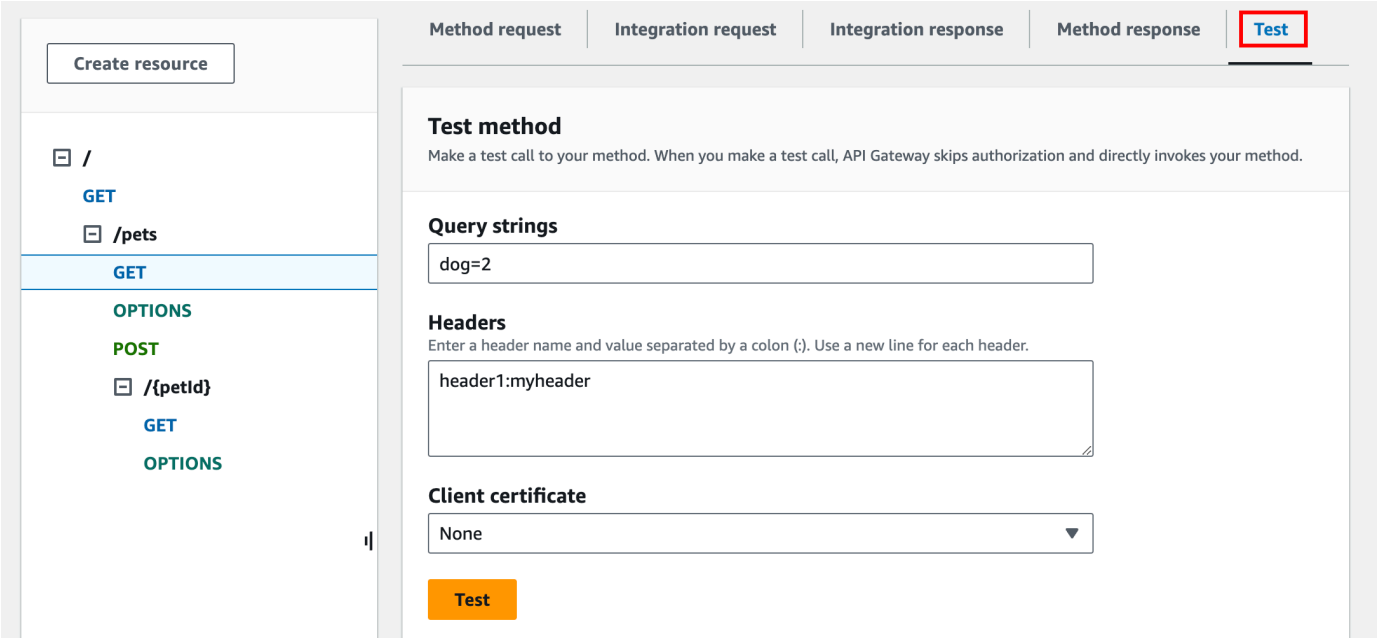
- È necessario specificare le impostazioni per i metodi che si intende testare. Segui le istruzioni in Metodi per REST APIs in API Gateway.

Test di un metodo tramite la console API Gateway

> **Important**
>
> Quando si testano i metodi con la console Gateway API è possibile che alle risorse vengano apportate modifiche non annullabili. Eseguire il test di un metodo tramite la console API Gateway equivale a chiamare il metodo dall'esterno della console. Ad esempio, se si utilizza la console API Gateway per chiamare un metodo che elimina le risorse di un'API e la chiamata del metodo riesce, le risorse vengono eliminate.

Test del metodo

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere un'API REST.
3. Nel riquadro Resources (Risorse) scegliere il metodo che si desidera testare.
4. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.



Immetti i valori nelle caselle visualizzate, ad esempio Stringhe di query, Intestazioni e Corpo della richiesta. La console include tali valori della richiesta del metodo nel modulo dell'applicazione/json predefinito.

Per specificare eventuali opzioni aggiuntive, contatta il proprietario dell'API.

5. Scegli Test (Esegui test). Verranno visualizzate le seguenti informazioni:

   - Request (Richiesta) è il percorso della risorsa chiamato per il metodo.
   - Status (Stato) è il codice dello stato HTTP della risposta.
   - La latenza (ms) è il tempo che intercorre tra la ricezione della richiesta dal chiamante e la risposta restituita.
   - Corpo della risposta è il corpo della risposta HTTP.
   - Intestazioni delle risposte sono le intestazioni di risposta HTTP.

   > **Tip**
   >
   > A seconda della mappatura, il codice di stato HTTP, il corpo della risposta e le intestazioni di risposta potrebbero essere diversi da quelli inviati dalla funzione Lambda, dal proxy HTTP o dal proxy di servizio. AWS

   - I log sono le voci simulate di Amazon CloudWatch Logs che sarebbero state scritte se questo metodo fosse stato chiamato al di fuori della console API Gateway.

> **Note**
>
> Sebbene le voci di CloudWatch Logs siano simulate, i risultati della chiamata al metodo sono reali.

Oltre a utilizzare la console API Gateway, puoi utilizzare AWS CLI o un AWS SDK per API Gateway per testare l'invocazione di un metodo. Per farlo utilizzando AWS CLI, vedi. test-invoke-method

Utilizzo di un SDK Java generato da API Gateway per un'API REST

In questa sezione verranno illustrate, a titolo di esempio, le fasi necessarie per utilizzare un SDK Java generato da API Gateway per un'API REST mediante l'API del calcolatore semplice. Per poter procedere, è necessario avere completato le fasi descritte in Genera SDKs per REST APIs in API Gateway.

Per installare e utilizzare l'SDK Java generato da API Gateway

1. Estrai il contenuto del file .zip generato da API Gateway scaricato precedentemente.
2. Scaricare e installare Apache Maven (versione 3.5 o successiva).
3. Scaricare e installare JDK 8.
4. Imposta la variabile di ambiente `JAVA_HOME`.
5. Passa alla cartella dell'SDK decompressa in cui si trova il file pom.xml. Per impostazione predefinita, questa cartella è `generated-code`. Eseguire il comando mvn install per installare i file di artefatto compilati nel repository Maven locale. Viene creata una cartella `target` contenente la libreria SDK compilata.
6. Digita il comando seguente in una directory vuota per creare uno stub di progetto del client e chiamare l'API mediante la libreria SDK installata.

```
mvn -B archetype:generate \
    -DarchetypeGroupdId=org.apache.maven.archetypes \
    -DgroupId=examples.aws.apig.simpleCalc.sdk.app \
    -DartifactId=SimpleCalc-sdkClient
```

> **Note**
>
> Nel comando precedente è stato inserito il separatore \ per maggiore leggibilità. Il comando deve trovarsi su un'unica riga senza il separatore.

Questo comando crea uno stub di applicazione. Lo stub dell'applicazione contiene un `pom.xml` file e una `src` cartella nella directory principale del progetto (*`SimpleCalc-SdkClient`* nel comando precedente). Inizialmente sono presenti due file di origine: `src/main/java/`*`{package-path}`*`/App.java` e `src/test/java/`*`{package-path}`*`/AppTest.java`. In questo esempio *`{package-path}`* è `examples/aws/apig/simpleCalc/sdk/app`. Il precorso del pacchetto viene derivato dal valore `DarchetypeGroupdId`. Puoi usare il file `App.java` come modello per l'applicazione client e, se necessario, aggiungerne altri nella stessa cartella. Puoi usare il file `AppTest.java` come modello di test di unità per l'applicazione e, se necessario, aggiungere altri file di codice di test nella stessa cartella.

7. Aggiorna le dipendenze di pacchetto nel file `pom.xml` generato come segue sostituendo, se necessario, le proprietà `groupId`, `artifactId`, `version` e `name` del progetto:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/
POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>examples.aws.apig.simpleCalc.sdk.app</groupId>
  <artifactId>SimpleCalc-sdkClient</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>SimpleCalc-sdkClient</name>
  <url>http://maven.apache.org</url>

   <dependencies>
      <dependency>
          <groupId>com.amazonaws</groupId>
          <artifactId>aws-java-sdk-core</artifactId>
          <version>1.11.94</version>
      </dependency>
      <dependency>
          <groupId>my-apig-api-examples</groupId>
          <artifactId>simple-calc-sdk</artifactId>
          <version>1.0.0</version>
```

```
      </dependency>

    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.12</version>
      <scope>test</scope>
    </dependency>

    <dependency>
        <groupId>commons-io</groupId>
        <artifactId>commons-io</artifactId>
        <version>2.5</version>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.5.1</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

> **Note**
>
> Quando una versione più recente dell'artefatto dipendente di `aws-java-sdk-core` è incompatibile con la versione specificata sopra (`1.11.94`), è necessario aggiornare il tag `<version>` alla nuova versione.

8. Quindi mostreremo come chiamare l'API mediante l'SDK chiamando i metodi `getABOp(GetABOpRequest req)`, `getApiRoot(GetApiRootRequest req)` e `postApiRoot(PostApiRootRequest req)` dell'SDK. Questi metodi corrispondono ai metodi

`GET /{a}/{b}/{op}`, `GET /?a={x}&b={y}&op={operator}` e `POST /`, rispettivamente con un payload di richieste API `{"a": x, "b": y, "op": "operator"}`.

Aggiorna il file `App.java` come segue:

```
package examples.aws.apig.simpleCalc.sdk.app;

import java.io.IOException;

import com.amazonaws.opensdk.config.ConnectionConfiguration;
import com.amazonaws.opensdk.config.TimeoutConfiguration;

import examples.aws.apig.simpleCalc.sdk.*;
import examples.aws.apig.simpleCalc.sdk.model.*;
import examples.aws.apig.simpleCalc.sdk.SimpleCalcSdk.*;

public class App
{
    SimpleCalcSdk sdkClient;

    public App() {
        initSdk();
    }

    // The configuration settings are for illustration purposes and may not be a
 recommended best practice.
    private void initSdk() {
        sdkClient = SimpleCalcSdk.builder()
              .connectionConfiguration(
                  new ConnectionConfiguration()
                        .maxConnections(100)
                        .connectionMaxIdleMillis(1000))
              .timeoutConfiguration(
                  new TimeoutConfiguration()
                        .httpRequestTimeout(3000)
                        .totalExecutionTimeout(10000)
                        .socketTimeout(2000))
        .build();

    }
    // Calling shutdown is not necessary unless you want to exert explicit control
 of this resource.
    public void shutdown() {
```

```
        sdkClient.shutdown();
    }

    // GetABOpResult getABOp(GetABOpRequest getABOpRequest)
    public Output getResultWithPathParameters(String x, String y, String operator)
{
     operator = operator.equals("+") ? "add" : operator;
     operator = operator.equals("/") ? "div" : operator;

        GetABOpResult abopResult = sdkClient.getABOp(new
GetABOpRequest().a(x).b(y).op(operator));
        return abopResult.getResult().getOutput();
    }

    public Output getResultWithQueryParameters(String a, String b, String op) {
        GetApiRootResult rootResult = sdkClient.getApiRoot(new
GetApiRootRequest().a(a).b(b).op(op));
        return rootResult.getResult().getOutput();
    }

    public Output getResultByPostInputBody(Double x, Double y, String o) {
     PostApiRootResult postResult = sdkClient.postApiRoot(
      new PostApiRootRequest().input(new Input().a(x).b(y).op(o)));
     return postResult.getResult().getOutput();
    }

    public static void main( String[] args )
    {
        System.out.println( "Simple calc" );
        // to begin
        App calc = new App();

        // call the SimpleCalc API
        Output res = calc.getResultWithPathParameters("1", "2", "-");
        System.out.printf("GET /1/2/-: %s\n", res.getC());

        // Use the type query parameter
        res = calc.getResultWithQueryParameters("1", "2", "+");
        System.out.printf("GET /?a=1&b=2&op=+: %s\n", res.getC());

        // Call POST with an Input body.
        res = calc.getResultByPostInputBody(1.0, 2.0, "*");
        System.out.printf("PUT /\n\n{\"a\":1, \"b\":2,\"op\":\"*\"}\n %s\n",
res.getC());
```

```
    }
}
```

Nel'esempio precedente le impostazioni di configurazione utilizzate per creare istanze del client SDK hanno esclusivamente scopo illustrativo e non sono necessariamente best practice consigliate. Anche la chiamata a `sdkClient.shutdown()` è facoltativa, soprattutto se si ha necessità di esercitare un controllo accurato su quando liberare risorse.

Abbiamo mostrato i modelli fondamentali per chiamare un'API utilizzando un SDK Java. Le istruzioni possono essere applicate per chiamare altri metodi API.

Utilizzo di un SDK Android generato da API Gateway per un'API REST

In questa sezione verranno illustrate le fasi necessarie per utilizzare un SDK Android generato da API Gateway per un'API REST. Per poter procedere, è necessario avere già completato le fasi descritte in Genera SDKs per REST APIs in API Gateway.

> **Note**
>
> L'SDK generato non è compatibile con Android 4.4 e versioni precedenti. Per ulteriori informazioni, consulta the section called “Note importanti”.

Per installare e utilizzare l'SDK Android generato da API Gateway

1. Estrai il contenuto del file .zip generato da API Gateway scaricato precedentemente.
2. Scaricare e installare Apache Maven (preferibilmente versione 3.x).
3. Scaricare e installare JDK 8.
4. Imposta la variabile di ambiente `JAVA_HOME`.
5. Eseguire il comando mvn install per installare i file di artefatto compilati nel repository Maven locale. Viene creata una cartella `target` contenente la libreria SDK compilata.
6. Copiare il file SDK (il cui nome viene derivato dai valori specificati per Artifact Id (Id artefatto) e Artifact Version (Versione artefatto) durante la generazione dell'SDK, ad esempio `simple-calcsdk-1.0.0.jar`) dalla cartella `target`, insieme a tutte le altre librerie della cartella `target/lib`, nella cartella `lib` del progetto.

Se usi Android Studio, crea una cartella `libs` nel modulo della tua app client e copia il file .jar richiesto in questa cartella. Verifica che la sezione delle dipendenze nel file gradle del modulo contenga il codice seguente.

```
    compile fileTree(include: ['*.jar'], dir: 'libs')
    compile fileTree(include: ['*.jar'], dir: 'app/libs')
```

Assicurati che non siano dichiarati file .jar duplicati.

7. Usa la classe `ApiClientFactory` per inizializzare l'SDK generato da API Gateway. Per esempio:

```
ApiClientFactory factory = new ApiClientFactory();

// Create an instance of your SDK. Here, 'SimpleCalcClient.java' is the compiled
 java class for the SDK generated by API Gateway.
final SimpleCalcClient client = factory.build(SimpleCalcClient.class);

// Invoke a method:
//   For the 'GET /?a=1&b=2&op=+' method exposed by the API, you can invoke it by
 calling the following SDK method:

Result output = client.rootGet("1", "2", "+");

//     where the Result class of the SDK corresponds to the Result model of the
 API.
//

//   For the 'GET /{a}/{b}/{op}'  method exposed by the API, you can call the
 following SDK method to invoke the request,

Result output = client.aBOpGet(a, b, c);

//     where a, b, c can be "1", "2", "add", respectively.

//   For the following API method:
//        POST /
//        host: ...
//        Content-Type: application/json
//
//        { "a": 1, "b": 2, "op": "+" }
// you can call invoke it by calling the rootPost method of the SDK as follows:
```

```
Input body = new Input();
input.a=1;
input.b=2;
input.op="+";
Result output = client.rootPost(body);

//      where the Input class of the SDK corresponds to the Input model of the API.

// Parse the result:
//     If the 'Result' object is { "a": 1, "b": 2, "op": "add", "c":3"}, you
 retrieve the result 'c') as

String result=output.c;
```

8. Per utilizzare un provider di credenziali Amazon Cognito per autorizzare le chiamate alla tua API, usa la `ApiClientFactory` classe per passare un set di AWS credenziali utilizzando l'SDK generato da API Gateway, come mostrato nell'esempio seguente.

```
// Use CognitoCachingCredentialsProvider to provide AWS credentials
// for the ApiClientFactory
AWSCredentialsProvider credentialsProvider = new CognitoCachingCredentialsProvider(
        context,          // activity context
        "identityPoolId", // Cognito identity pool id
        Regions.US_EAST_1 // region of Cognito identity pool
);

ApiClientFactory factory = new ApiClientFactory()
  .credentialsProvider(credentialsProvider);
```

9. Per impostare una chiave API utilizzando l'SDK generato da API Gateway, usa un codice simile al seguente.

```
ApiClientFactory factory = new ApiClientFactory()
  .apiKey("YOUR_API_KEY");
```

Usa un file JavaScript SDK generato da API Gateway per un REST API

La procedura seguente mostra come utilizzare un file JavaScript SDK generato da API Gateway.

> **Note**
>
> Queste istruzioni presuppongono che tu abbia già completato la procedura descritta in Genera SDKs per REST APIs in API Gateway.

> **Important**
>
> Se sono stati definiti API solo ANY metodi, il SDK pacchetto generato non conterrà alcun `apigClient.js` file e sarà necessario definire i ANY metodi autonomamente.

Per installare, avviare e richiamare un file JavaScript SDK generato da API Gateway for a REST API

1. Estrai il contenuto del file.zip API generato dal Gateway che hai scaricato in precedenza.
2. Abilita la condivisione delle risorse tra le origini (CORS) per tutti i metodi che verranno richiamati dal Gateway SDK generato. API Per istruzioni, consulta CORSper REST APIs in API Gateway.
3. Nella tua pagina Web, includi i riferimenti ai seguenti script.

```
<script type="text/javascript" src="lib/axios/dist/axios.standalone.js"></script>
<script type="text/javascript" src="lib/CryptoJS/rollups/hmac-sha256.js"></script>
<script type="text/javascript" src="lib/CryptoJS/rollups/sha256.js"></script>
<script type="text/javascript" src="lib/CryptoJS/components/hmac.js"></script>
<script type="text/javascript" src="lib/CryptoJS/components/enc-base64.js"></
script>
<script type="text/javascript" src="lib/url-template/url-template.js"></script>
<script type="text/javascript" src="lib/apiGatewayCore/sigV4Client.js"></script>
<script type="text/javascript" src="lib/apiGatewayCore/apiGatewayClient.js"></
script>
<script type="text/javascript" src="lib/apiGatewayCore/simpleHttpClient.js"></
script>
<script type="text/javascript" src="lib/apiGatewayCore/utils.js"></script>
<script type="text/javascript" src="apigClient.js"></script>
```

4. Nel codice, inizializza quello SDK generato da API Gateway utilizzando un codice simile al seguente.

```
var apigClient = apigClientFactory.newClient();
```

Per inizializzare il codice SDK generato da API Gateway con AWS le credenziali, utilizzate un codice simile al seguente. Se si utilizzano AWS le credenziali, tutte le richieste API verranno firmate.

```
var apigClient = apigClientFactory.newClient({
  accessKey: 'ACCESS_KEY',
  secretKey: 'SECRET_KEY',
});
```

Per utilizzare una API chiave con quella SDK generata da API Gateway, passatela API come parametro all'`Factory`oggetto utilizzando un codice simile al seguente. Se usi una API chiave, questa viene specificata come parte dell'`x-api-key`intestazione e tutte le richieste API verranno firmate. Ciò significa che è necessario impostare le intestazioni CORS Accept appropriate per ogni richiesta.

```
var apigClient = apigClientFactory.newClient({
  apiKey: 'API_KEY'
});
```

5. Chiama i API metodi in API Gateway utilizzando un codice simile al seguente. Ogni chiamata restituisce una promessa con callback di successo e di fallimento.

```
var params = {
  // This is where any modeled request parameters should be added.
  // The key is the parameter name, as it is defined in the API in API Gateway.
  param0: '',
  param1: ''
};

var body = {
  // This is where you define the body of the request,
};

var additionalParams = {
  // If there are any unmodeled query parameters or headers that must be
  //   sent with the request, add them here.
  headers: {
```

```
    param0: '',
    param1: ''
  },
  queryParams: {
    param0: '',
    param1: ''
  }
};

apigClient.methodName(params, body, additionalParams)
    .then(function(result){
      // Add success callback code here.
    }).catch( function(result){
      // Add error callback code here.
    });
```

Ecco, il *methodName* è costruito dal percorso delle risorse della richiesta del metodo e dal HTTP verbo. Per SimpleCalc API, i SDK metodi per i API metodi di

```
1. GET /?a=...&b=...&op=...
2. POST /

   { "a": ..., "b": ..., "op": ...}
3. GET /{a}/{b}/{op}
```

i SDK metodi corrispondenti sono i seguenti:

```
1. rootGet(params);      // where params={"a": ..., "b": ..., "op": ...} is
 resolved to the query parameters
2. rootPost(null, body); // where body={"a": ..., "b": ..., "op": ...}
3. aBOpGet(params);      // where params={"a": ..., "b": ..., "op": ...} is
 resolved to the path parameters
```

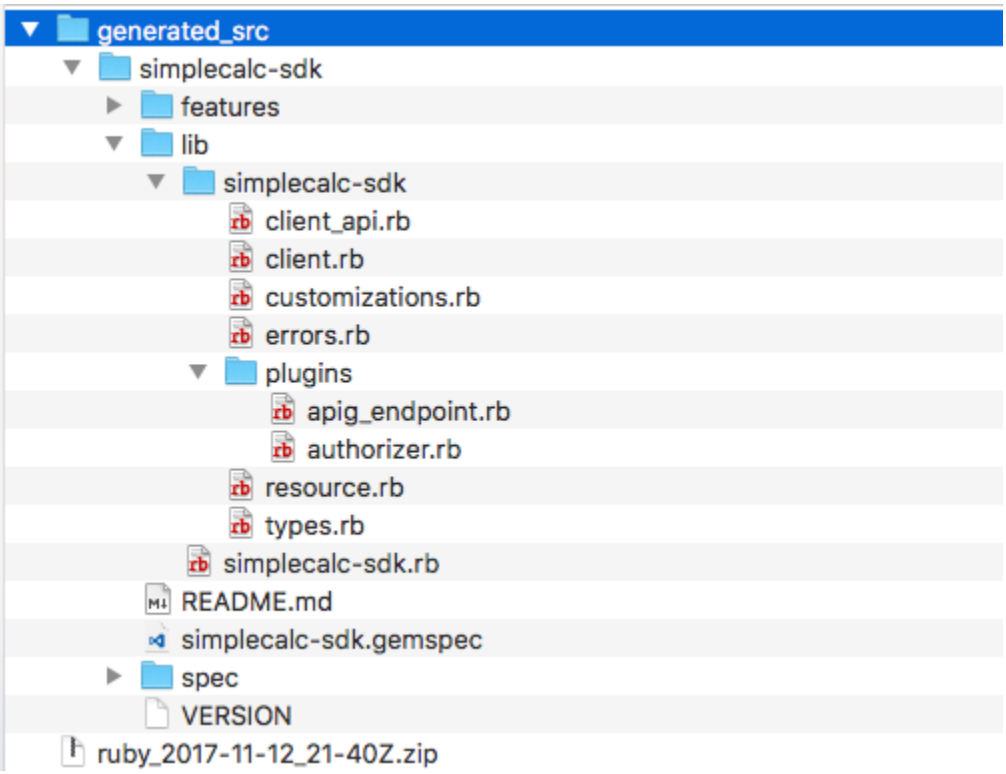
Usa un Ruby SDK generato da API Gateway per un REST API

La procedura seguente mostra come utilizzare un Ruby SDK API generato da Gateway.

> **Note**
>
> Queste istruzioni presuppongono che sia già stata completata la procedura in Genera SDKs per REST APIs in API Gateway.

Per installare, creare un'istanza e chiamare un Ruby SDK generato da Gateway per un API REST API

1. Decomprimi il file Ruby scaricato. SDK La SDK fonte generata è mostrata come segue.



2. Crea un Ruby Gem dalla SDK fonte generata, usando i seguenti comandi di shell in una finestra di terminale:

```
# change to /simplecalc-sdk directory
cd simplecalc-sdk

# build the generated gem
gem build simplecalc-sdk.gemspec
```

   Dopo questa operazione, simplecalc-sdk-1.0.0.gem diventa disponibile.

3. Installa la gem:

```
gem install simplecalc-sdk-1.0.0.gem
```

4. Crea un'applicazione client. Crea un'istanza e inizializza il client Ruby SDK nell'app:

```
require 'simplecalc-sdk'
client = SimpleCalc::Client.new(
    http_wire_trace: true,
    retry_limit: 5,
    http_read_timeout: 50
)
```

Se è configurata l'autorizzazione API has del `AWS_IAM` tipo, puoi includere le AWS credenziali del chiamante fornendo e durante l'inizializzazione: `accessKey` `secretKey`

```
require 'pet-sdk'
client = Pet::Client.new(
    http_wire_trace: true,
    retry_limit: 5,
    http_read_timeout: 50,
    access_key_id: 'ACCESS_KEY',
    secret_access_key: 'SECRET_KEY'
)
```

5. Effettua API chiamate tramite l'app. SDK

> **Tip**
>
> Se non conosci le convenzioni relative alle chiamate ai SDK metodi, puoi esaminare il `client.rb` file nella SDK `lib` cartella generata. La cartella contiene la documentazione di ogni chiamata al API metodo supportata.

Per conoscere le operazioni supportate:

```
# to show supported operations:
puts client.operation_names
```

Ciò si traduce nella seguente visualizzazione, corrispondente rispettivamente ai API metodi e `GET /{a}/{b}/{op}``POST /`, più un payload del `{a:"…", b:"…", op:"…"}` formato: `GET /?a={.}&b={.}&op={.}`

```
[:get_api_root, :get_ab_op, :post_api_root]
```

Per richiamare il `GET /?a=1&b=2&op=+` API metodo, chiamate il seguente metodo RubySDK:

```
resp = client.get_api_root({a:"1", b:"2", op:"+"})
```

Per richiamare il `POST /` API metodo con un payload di`{a: "1", b: "2", "op": "+"}`, chiamate il seguente metodo Ruby: SDK

```
resp = client.post_api_root(input: {a:"1", b:"2", op:"+"})
```

Per richiamare il `GET /1/2/+` API metodo, chiamate il seguente metodo Ruby: SDK

```
resp = client.get_ab_op({a:"1", b:"2", op:"+"})
```

Le chiamate al SDK metodo riuscite restituiscono la seguente risposta:

```
resp : {
    result: {
        input: {
            a: 1,
            b: 2,
            op: "+"
        },
        output: {
            c: 3
        }
    }
}
```

Uso dell'SDK iOS generato da API Gateway per un'API REST in Objective-C o Swift

In questo tutorial mostreremo come usare un SDK iOS generato da API Gateway per un'API REST in un'app Objective-C o Swift per chiamare l'API sottostante. Useremo l'SimpleCalc API come esempio per illustrare i seguenti argomenti:

- Come installare i componenti AWS Mobile SDK richiesti nel tuo progetto Xcode
- Come creare l'oggetto client dell'API prima di chiamare i metodi dell'API
- Come chiamare i metodi API attraverso i metodi SDK corrispondenti nell'oggetto client dell'API
- Come preparare un input di metodo e analizzare il risultato utilizzando le classi di modello corrispondenti dell'SDK

Argomenti

- Utilizzo di un SDK iOS (Objective-C) generato per chiamare l'API
- Utilizzo di un SDK iOS (Swift) generato per chiamare l'API

Utilizzo di un SDK iOS (Objective-C) generato per chiamare l'API

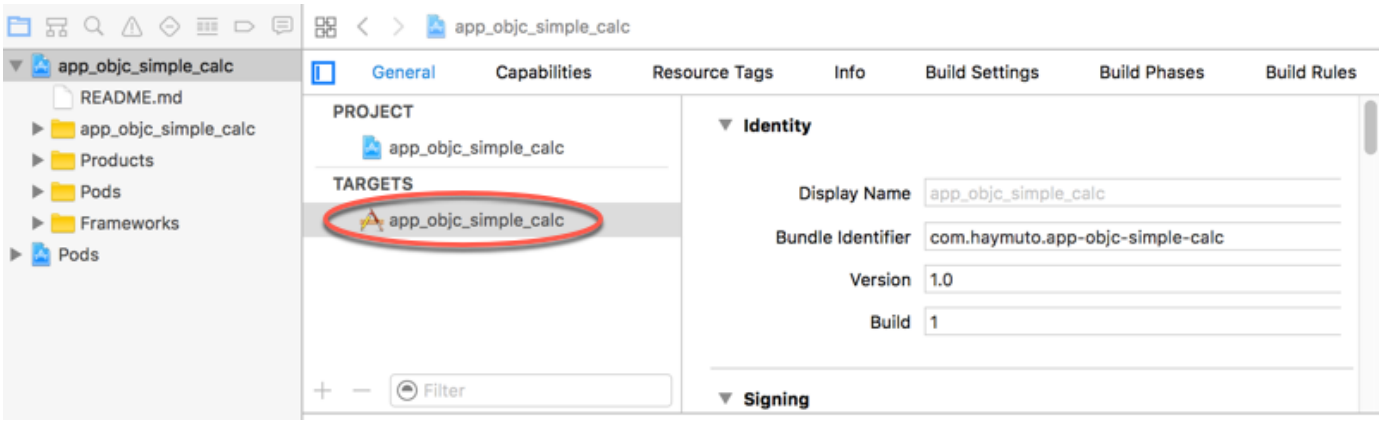
Prima di iniziare la procedura seguente, è necessario completare le fasi descritte in Genera SDKs per REST APIs in API Gateway per iOS in Objective-C e scaricare il file .zip dell'SDK generato.

Installa l'SDK AWS mobile e un SDK iOS generato da API Gateway in un progetto Objective-C

La procedura seguente descrive come installare l'SDK.

Per installare e utilizzare un SDK iOS generato da API Gateway in Objective-C

1. Estrai il contenuto del file .zip generato da API Gateway scaricato precedentemente. Utilizzando l'SimpleCalc API, potresti voler rinominare la cartella SDK decompressa con qualcosa del genere. **`sdk_objc_simple_calc`** In questa cartella SDK sono presenti un file `README.md` e un file `Podfile`. Il file `README.md` contiene le istruzioni per installare e usare l'SDK. Questo tutorial fornisce i dettagli relativi alle istruzioni. L'installazione consente di CocoaPodsimportare le librerie API Gateway richieste e altri componenti AWS Mobile SDK dipendenti. Per importare gli SDK nel progetto Xcode dell'app, devi aggiornare il `Podfile`. La cartella SDK non archiviata contiene anche una cartella `generated-src` con il codice sorgente dell'SDK generato dell'API.
2. Avvia Xcode e crea un nuovo progetto Objective-C per iOS. Prendi nota della destinazione del progetto. Dovrai specificare questa impostazione nel `Podfile`.



3. Per AWS Mobile SDK for iOS importarlo nel progetto Xcode utilizzando CocoaPods, procedi come segue:

   a. Installa CocoaPods eseguendo il seguente comando in una finestra di terminale:

   ```
   sudo gem install cocoapods
   pod setup
   ```

   b. Copia il file `Podfile` dalla cartella SDK estratta nella stessa directory in cui si trova il file di progetto Xcode. Sostituisci il blocco seguente:

   ```
   target '<YourXcodeTarget>' do
       pod 'AWSAPIGateway', '~> 2.4.7'
   end
   ```

   con il nome di destinazione del progetto:

   ```
   target 'app_objc_simple_calc' do
       pod 'AWSAPIGateway', '~> 2.4.7'
   end
   ```

   Se il progetto Xcode già contiene un file denominato `Podfile`, aggiungi la riga di codice seguente:

   ```
   pod 'AWSAPIGateway', '~> 2.4.7'
   ```

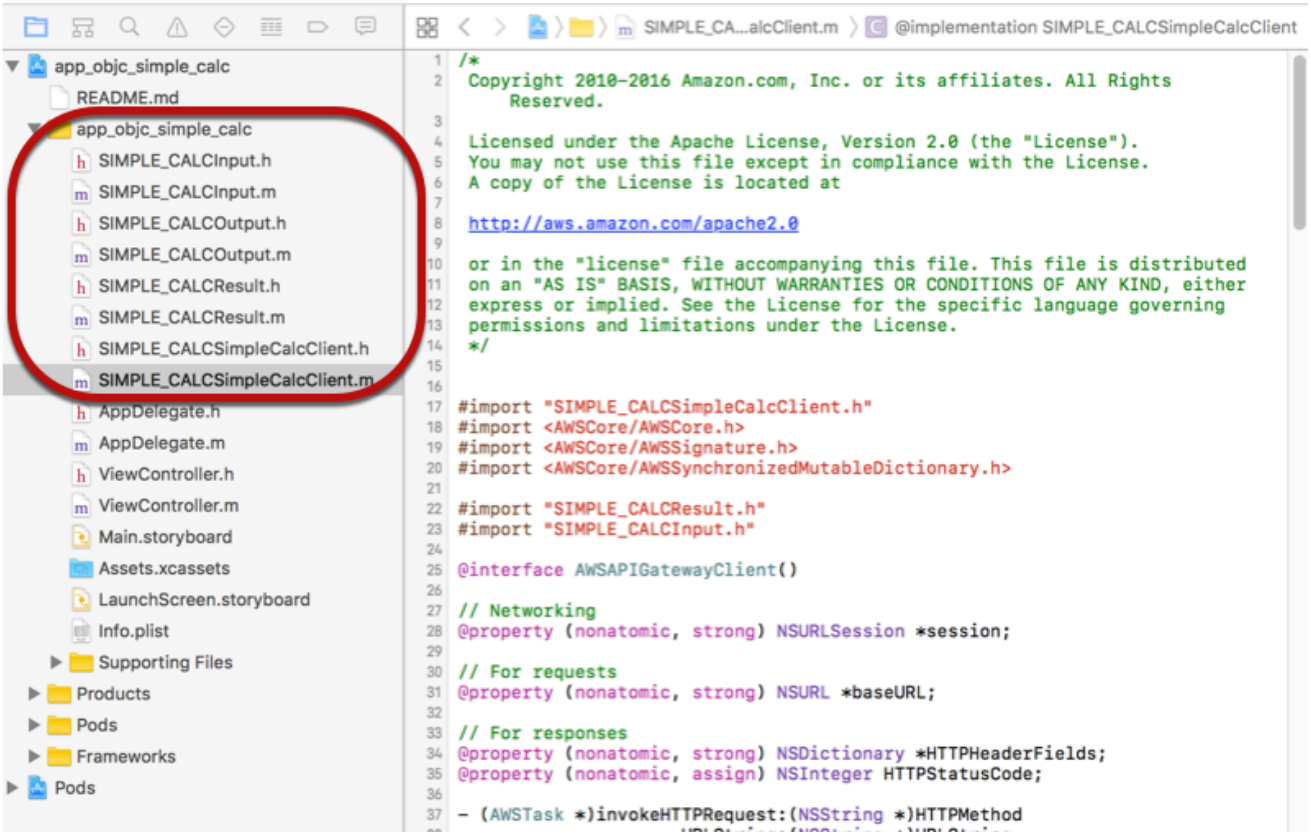
   c. Apri una finestra del terminale ed esegui il comando seguente:

   ```
   pod install
   ```

Questo installa il componente API Gateway e altri componenti AWS Mobile SDK dipendenti.

d. Chiudi il progetto Xcode e apri il file `.xcworkspace` per riavviare Xcode.

e. Aggiungi tutti i file `.h` e `.m` dalla directory `generated-src` estratta dell'SDK nel progetto Xcode.



Per importare AWS Mobile SDK for iOS Objective-C nel tuo progetto scaricando esplicitamente AWS Mobile SDK o usando Carthage, segui le istruzioni nel file README.md. Assicurati di utilizzare solo una di queste opzioni per importare Mobile SDK. AWS

Chiamata ai metodi API mediante l'SDK iOS generato da API Gateway in un progetto Objective-C

Quando avete generato l'SDK con il prefisso di `SIMPLE_CALC` for questa SimpleCalc API con due modelli per l'input (`Input`) e l'output (`Result`) dei metodi, nell'SDK, la classe client API risultante diventa `SIMPLE_CALCSimpleCalcClient` e le classi di dati corrispondenti sono `SIMPLE_CALCInput` e, rispettivamente. `SIMPLE_CALCResult` Le richieste e le risposte API sono mappate ai metodi SDK come segue:

- La richiesta API di

  ```
  GET /?a=...&b=...&op=...
  ```

  diventa il metodo SDK di

  ```
  (AWSTask *)rootGet:(NSString *)op a:(NSString *)a b:(NSString *)b
  ```

  La proprietà `AWSTask.result` è del tipo `SIMPLE_CALCResult`, se il modello `Result` è stato aggiunto alla risposta del metodo. In caso contrario, è del tipo `NSDictionary`.

- Questa richiesta API di

  ```
  POST /

  {
     "a": "Number",
     "b": "Number",
     "op": "String"
  }
  ```

  diventa il metodo SDK di

  ```
  (AWSTask *)rootPost:(SIMPLE_CALCInput *)body
  ```

- La richiesta API di

  ```
  GET /{a}/{b}/{op}
  ```

  diventa il metodo SDK di

  ```
  (AWSTask *)aBOpGet:(NSString *)a b:(NSString *)b op:(NSString *)op
  ```

La procedura seguente descrive come chiamare i metodi API nel codice di origine delle app Objective-C, ad esempio nell'ambito del delegato `viewDidLoad` in un file `ViewController.m`.

Per chiamare l'API mediante l'SDK iOS generato da API Gateway

1. Importa il file di intestazione della classe client dell'API per rendere tale classe chiamabile nell'app:

```
#import "SIMPLE_CALCSimpleCalc.h"
```

   L'istruzione `#import` inoltre importa `SIMPLE_CALCInput.h` e `SIMPLE_CALCResult.h` per le due classi di modelli.

2. Crea un'istanza della classe client dell'API:

```
SIMPLE_CALCSimpleCalcClient *apiInstance = [SIMPLE_CALCSimpleCalcClient
 defaultClient];
```

   Per utilizzare Amazon Cognito con l'API, impostare la proprietà `defaultServiceConfiguration` sull'oggetto `AWSServiceManager` predefinito, come mostrato di seguito, prima di chiamare il metodo `defaultClient` per creare l'oggetto client dell'API (mostrato nell'esempio precedente):

```
AWSCognitoCredentialsProvider *creds = [[AWSCognitoCredentialsProvider alloc]
 initWithRegionType:AWSRegionUSEast1 identityPoolId:your_cognito_pool_id];
AWSServiceConfiguration *configuration = [[AWSServiceConfiguration alloc]
 initWithRegion:AWSRegionUSEast1 credentialsProvider:creds];
AWSServiceManager.defaultServiceManager.defaultServiceConfiguration =
 configuration;
```

3. Chiama il metodo `GET /?a=1&b=2&op=+` per eseguire `1+2`:

```
[[apiInstance rootGet: @"+" a:@"1" b:@"2"] continueWithBlock:^id _Nullable(AWSTask
 * _Nonnull task) {
    _textField1.text = [self handleApiResponse:task];
    return nil;
}];
```

   dove la funzione dell'helper `handleApiResponse:task` formatta il risultato come stringa da visualizzare in un campo di testo (`_textField1`).

```
- (NSString *)handleApiResponse:(AWSTask *)task {
    if (task.error != nil) {
```

```
        return [NSString stringWithFormat: @"Error: %@", task.error.description];
    } else if (task.result != nil && [task.result isKindOfClass:[SIMPLE_CALCResult
 class]]) {
        return [NSString stringWithFormat:@"%@ %@ %@ = %@\n",task.result.input.a,
 task.result.input.op, task.result.input.b, task.result.output.c];
    }
    return nil;
}
```

Il risultato visualizzato è `1 + 2 = 3`.

4. Chiama `POST /` con un payload per eseguire `1-2`:

```
SIMPLE_CALCInput *input = [[SIMPLE_CALCInput alloc] init];
    input.a = [NSNumber numberWithInt:1];
    input.b = [NSNumber numberWithInt:2];
    input.op = @"-";
    [[apiInstance rootPost:input] continueWithBlock:^id _Nullable(AWSTask *
 _Nonnull task) {
        _textField2.text = [self handleApiResponse:task];
        return nil;
    }];
```

Il risultato visualizzato è `1 - 2 = -1`.

5. Chiama `GET /{a}/{b}/{op}` per eseguire `1/2`:

```
[[apiInstance aBOpGet:@"1" b:@"2" op:@"div"] continueWithBlock:^id
 _Nullable(AWSTask * _Nonnull task) {
    _textField3.text = [self handleApiResponse:task];
    return nil;
}];
```

Il risultato visualizzato è `1 div 2 = 0.5`. Nell'esempio, `div` viene usato al posto di `/` perché la funzione Lambda semplice nel back-end non gestisce le variabili di percorso con codifica URL.

Utilizzo di un SDK iOS (Swift) generato per chiamare l'API

Prima di iniziare la procedura seguente, devi completare le fasi descritte in Genera SDKs per REST APIs in API Gateway per iOS in Swift e scaricare il file .zip dell'SDK generato.

Argomenti

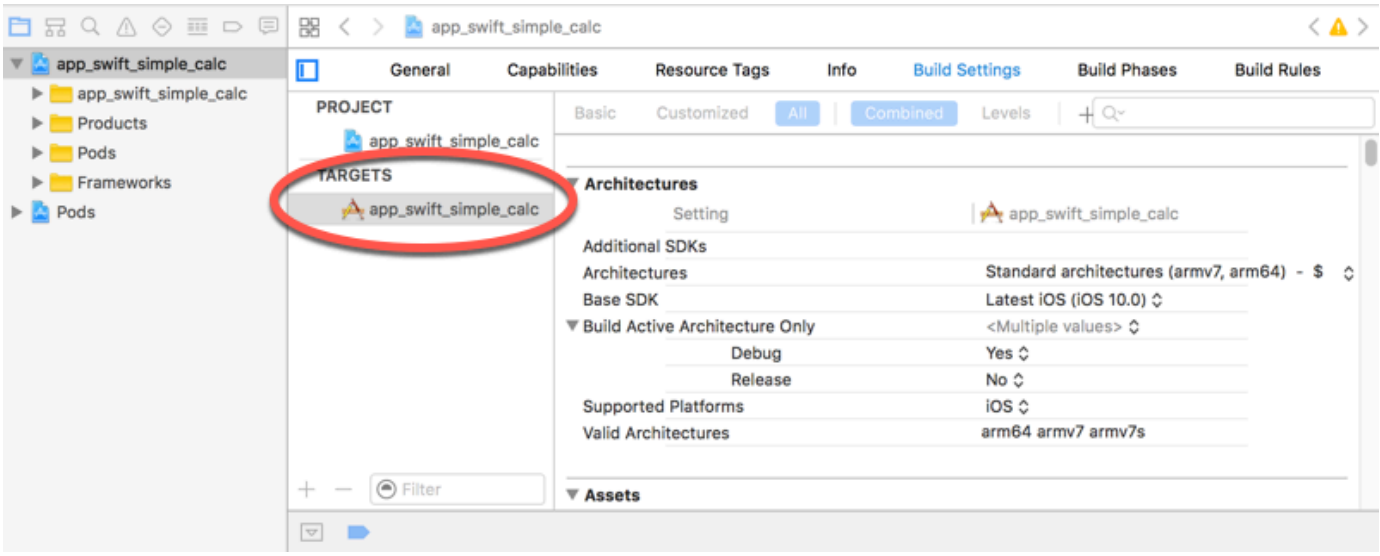
- Installa SDK AWS mobile e SDK generato da API Gateway in un progetto Swift
- Chiamata ai metodi API mediante l'SDK iOS generato da API Gateway in un progetto Swift

Installa SDK AWS mobile e SDK generato da API Gateway in un progetto Swift

La procedura seguente descrive come installare l'SDK.

Per installare e utilizzare un SDK iOS generato da API Gateway in Swift

1. Estrai il contenuto del file .zip generato da API Gateway scaricato precedentemente. Utilizzando l'SimpleCalc API, potresti voler rinominare la cartella SDK decompressa con qualcosa del genere. **`sdk_swift_simple_calc`** In questa cartella SDK sono presenti un file `README.md` e un file `Podfile`. Il file `README.md` contiene le istruzioni per installare e usare l'SDK. Questo tutorial fornisce i dettagli relativi alle istruzioni. L'installazione consente di importare i componenti necessari CocoaPodsdi Mobile SDK. AWS Per importare gli SDK nel progetto Xcode dell'app Swift, devi aggiornare il `Podfile`. La cartella SDK non archiviata contiene anche una cartella `generated-src` con il codice sorgente dell'SDK generato dell'API.
2. Avvia Xcode e crea un nuovo progetto Swift per iOS. Prendi nota della destinazione del progetto. Dovrai specificare questa impostazione nel `Podfile`.



3. Per importare i componenti AWS Mobile SDK richiesti nel progetto Xcode utilizzando CocoaPods, procedi come segue:

   a. Se non è installato, installalo CocoaPods eseguendo il seguente comando in una finestra di terminale:

```
sudo gem install cocoapods
pod setup
```

b. Copia il file `Podfile` dalla cartella SDK estratta nella stessa directory in cui si trova il file di progetto Xcode. Sostituisci il blocco seguente:

```
target '<YourXcodeTarget>' do
    pod 'AWSAPIGateway', '~> 2.4.7'
end
```

con il nome di destinazione del progetto, come mostrato

```
target 'app_swift_simple_calc' do
    pod 'AWSAPIGateway', '~> 2.4.7'
end
```

Se il progetto Xcode già contiene un `Podfile` con la destinazione corretta, puoi semplicemente aggiungere la riga di codice seguente al loop `do ... end`:

```
pod 'AWSAPIGateway', '~> 2.4.7'
```

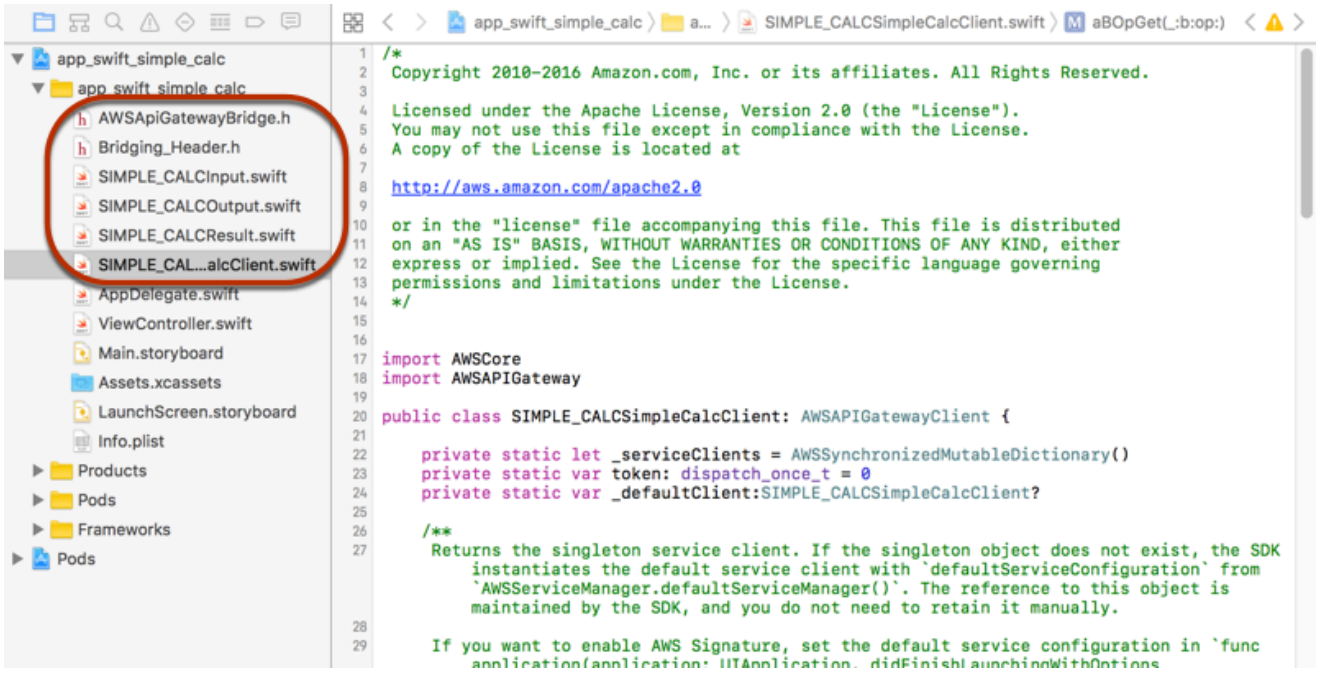
c. Apri una finestra del terminale ed esegui il comando seguente nella directory dell'app:

```
pod install
```

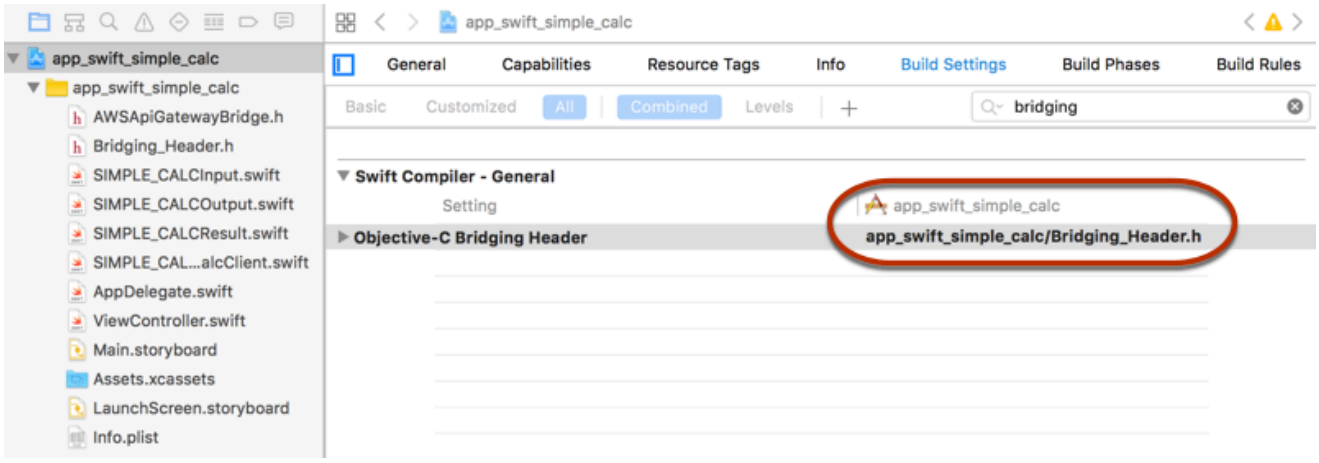
Questo installa il componente API Gateway e tutti i componenti AWS Mobile SDK dipendenti nel progetto dell'app.

d. Chiudi il progetto Xcode e apri il file `*.xcworkspace` per riavviare Xcode.

e. Aggiungi tutti i file di intestazione dell'SDK (`.h`) e i file del codice di origine Swift (`.swift`) dalla directory `generated-src` estratta nel progetto Xcode.



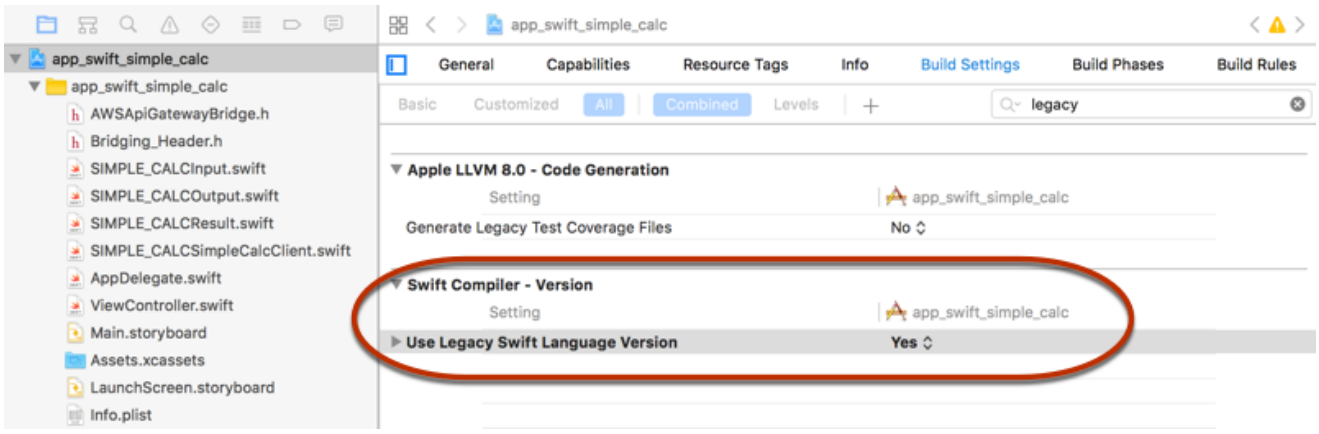
f. Per abilitare la chiamata alle librerie Objective-C di AWS Mobile SDK dal tuo progetto di codice Swift, imposta il percorso del **`Bridging_Header.h`** file nella proprietà Objective-C Bridging Header in Swift Compiler - Impostazione generale della configurazione del tuo progetto Xcode:



> **Tip**
>
> È possibile digitare **`bridging`** nella casella di ricerca di Xcode per trovare la proprietà Objective-C Bridging Header (Intestazione bridging Objective-C).

g. Prima di procedere, crea il progetto Xcode per verificare che sia configurato correttamente. Se il tuo Xcode utilizza una versione di Swift più recente di quella supportata per Mobile

SDK, otterrai errori nel compilatore Swift. AWS In questo caso, impostare la proprietà Use Legacy Swift Language Version (Usa versione linguaggio Swift legacy) su Yes (Sì) in Swift Compiler - Version (Compilatore Swift - Versione):



Per importare AWS Mobile SDK for iOS in Swift nel tuo progetto AWS scaricando esplicitamente Mobile SDK o utilizzando Carthage, segui le istruzioni nel `README.md` file fornito con il pacchetto SDK. Assicurati di utilizzare solo una di queste opzioni per importare Mobile SDK. AWS

Chiamata ai metodi API mediante l'SDK iOS generato da API Gateway in un progetto Swift

Quando avete generato l'SDK con il prefisso di `SIMPLE_CALC` for questa SimpleCalc API con due modelli per descrivere l'input (`Input`) e l'output (`Result`) delle richieste e delle risposte dell'API, nell'SDK, la classe client API risultante diventa `SIMPLE_CALCSimpleCalcClient` e le classi di dati corrispondenti sono `SIMPLE_CALCInput` e, rispettivamente. `SIMPLE_CALCResult` Le richieste e le risposte API sono mappate ai metodi SDK come segue:

- La richiesta API di

  ```
  GET /?a=...&b=...&op=...
  ```

  diventa il metodo SDK di

  ```
  public func rootGet(op: String?, a: String?, b: String?) -> AWSTask
  ```

  La proprietà `AWSTask.result` è del tipo `SIMPLE_CALCResult`, se il modello `Result` è stato aggiunto alla risposta del metodo. In caso contrario, è del tipo `NSDictionary`.

- Questa richiesta API di

```
POST /

{
   "a": "Number",
   "b": "Number",
   "op": "String"
}
```

diventa il metodo SDK di

```
public func rootPost(body: SIMPLE_CALCInput) -> AWSTask
```

- La richiesta API di

```
GET /{a}/{b}/{op}
```

diventa il metodo SDK di

```
public func aBOpGet(a: String, b: String, op: String) -> AWSTask
```

La procedura seguente descrive come chiamare i metodi API nel codice di origine delle app Swift, ad esempio nell'ambito del delegato `viewDidLoad()` in un file `ViewController.m`.

Per chiamare l'API mediante l'SDK iOS generato da API Gateway

1. Crea un'istanza della classe client dell'API:

```
let client = SIMPLE_CALCSimpleCalcClient.default()
```

Per utilizzare Amazon Cognito con l'API, imposta una configurazione di AWS servizio predefinita (mostrata di seguito) prima di ottenere il `default` metodo (mostrato in precedenza):

```
let credentialsProvider =
 AWSCognitoCredentialsProvider(regionType: AWSRegionType.USEast1, identityPoolId:
 "my_pool_id")
let configuration = AWSServiceConfiguration(region: AWSRegionType.USEast1,
 credentialsProvider: credentialsProvider)
```

```
AWSServiceManager.defaultServiceManager().defaultServiceConfiguration =
 configuration
```

2. Chiama il metodo `GET /?a=1&b=2&op=+` per eseguire `1+2`:

```
client.rootGet("+", a: "1", b:"2").continueWithBlock {(task: AWSTask) -> AnyObject?
 in
    self.showResult(task)
    return nil
}
```

dove la funzione dell'helper `self.showResult(task)` stampa il risultato o l'errore nella console, ad esempio:

```
func showResult(task: AWSTask) {
    if let error = task.error {
        print("Error: \(error)")
    } else if let result = task.result {
        if result is SIMPLE_CALCResult {
            let res = result as! SIMPLE_CALCResult
            print(String(format:"%@ %@ %@ = %@", res.input!.a!, res.input!.op!,
 res.input!.b!, res.output!.c!))
        } else if result is NSDictionary {
            let res = result as! NSDictionary
            print("NSDictionary: \(res)")
        }
    }
}
```

In un'app di produzione puoi visualizzare il risultato o l'errore in un campo di testo. Il risultato visualizzato è `1 + 2 = 3`.

3. Chiama `POST /` con un payload per eseguire `1-2`:

```
let body = SIMPLE_CALCInput()
body.a=1
body.b=2
body.op="-"
client.rootPost(body).continueWithBlock {(task: AWSTask) -> AnyObject? in
    self.showResult(task)
    return nil
}
```

Il risultato visualizzato è `1 - 2 = -1`.

4. Chiama `GET /{a}/{b}/{op}` per eseguire `1/2`:

```
client.aBOpGet("1", b:"2", op:"div").continueWithBlock {(task: AWSTask) ->
 AnyObject? in
    self.showResult(task)
    return nil
}
```

Il risultato visualizzato è `1 div 2 = 0.5`. Nell'esempio, `div` viene usato al posto di `/` perché la funzione Lambda semplice nel back-end non gestisce le variabili di percorso con codifica URL.

Sviluppa REST APIs utilizzando Open API in API Gateway

È possibile utilizzare API Gateway per importare un file REST API di definizione esterno in API Gateway. Attualmente, API Gateway supporta i file di definizione OpenAPIOpenAPIv2.0 e v3.0, con le eccezioni elencate in. Note importanti su Amazon API Gateway per REST APIs È possibile aggiornare un API file sovrascrivendolo con una nuova definizione oppure unire una definizione con una esistente. API Le opzioni vengono specificate utilizzando un parametro di `mode` query nella richiesta. URL

Per un tutorial sull'utilizzo della API funzionalità di importazione dalla console API Gateway, consultaTutorial: creazione di un'API REST mediante l'importazione di un esempio.

Argomenti

- Importazione di un'API ottimizzata per l'edge in API Gateway
- Importazione di un'API regionale in API Gateway
- Importa un API file aperto per aggiornare una API definizione esistente
- Impostare la proprietà Open API basePath
- AWS variabili per Open import API
- Errori e avvisi derivanti dall'importazione in Gateway API API
- Esporta un file REST API da API Gateway

Importazione di un'API ottimizzata per l'edge in API Gateway

Puoi importare un file di definizione OpenAPI dell'API per creare una nuova API ottimizzata per gli edge specificando il tipo di endpoint `EDGE` come input aggiuntivo, oltre al file OpenAPI, nell'operazione di importazione. Puoi farlo utilizzando la console API Gateway o un AWS SDK. AWS CLI

Per un tutorial sull'utilizzo della caratteristica Importa API dalla console API Gateway, consulta Tutorial: creazione di un'API REST mediante l'importazione di un esempio.

Argomenti

- Importazione di un'API ottimizzata per l'edge mediante la console API Gateway
- Importa un'API ottimizzata per l'edge utilizzando il AWS CLI

Importazione di un'API ottimizzata per l'edge mediante la console API Gateway

Per importare un'API ottimizzata per l'edge tramite la console API Gateway, procedi nel seguente modo:

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Seleziona Create API (Crea API).
3. In API REST, scegliere Import (Importa).
4. Copia una definizione OpenAPI dell'API e incollala nell'editor di codice o seleziona Scegli il file per caricare un file OpenAPI da un'unità locale.
5. In Tipo di endpoint, scegli Ottimizzato per gli edge.
6. Scegli Crea API per iniziare a importare le definizioni OpenAPI.

Importa un'API ottimizzata per l'edge utilizzando il AWS CLI

Per importare un'API da un file di definizione OpenAPI per creare una nuova API ottimizzata per i bordi utilizzando il AWS CLI, utilizzare il comando come segue: `import-rest-api`

```
aws apigateway import-rest-api \
    --fail-on-warnings \
    --body 'file://path/to/API_OpenAPI_template.json'
```

oppure specificando esplicitamente il parametro della stringa di query `endpointConfigurationTypes` come `EDGE`:

```
aws apigateway import-rest-api \
    --parameters endpointConfigurationTypes=EDGE \
    --fail-on-warnings \
    --body 'file://path/to/API_OpenAPI_template.json'
```

Importazione di un'API regionale in API Gateway

Quando importi un'API, puoi scegliere la configurazione dell'endpoint regionale per l'API. Puoi utilizzare la console API Gateway AWS CLI, o un AWS SDK.

Quando esporti un'API, la configurazione dell'endpoint dell'API non è inclusa nelle definizioni dell'API esportate.

Per un tutorial sull'utilizzo della caratteristica Importa API dalla console API Gateway, consulta Tutorial: creazione di un'API REST mediante l'importazione di un esempio.

Argomenti

- Importazione di un'API regionale tramite la console API Gateway
- Importazione di un'API regionale tramite AWS CLI

Importazione di un'API regionale tramite la console API Gateway

Per importare un'API di un endpoint regionale mediante la console API Gateway, procedi come indicato di seguito:

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Seleziona Create API (Crea API).
3. In API REST, scegliere Import (Importa).
4. Copia una definizione OpenAPI dell'API e incollala nell'editor di codice o seleziona Scegli il file per caricare un file OpenAPI da un'unità locale.
5. In Tipo di endpoint, scegliRegionale.
6. Scegli Crea API per iniziare a importare le definizioni OpenAPI.

Importazione di un'API regionale tramite AWS CLI

Per importare un'API da un file di definizione OpenAPI utilizzando il AWS CLI, usa il `import-rest-api` comando:

```
aws apigateway import-rest-api \
    --parameters endpointConfigurationTypes=REGIONAL \
    --fail-on-warnings \
    --body 'file://path/to/API_OpenAPI_template.json'
```

Importa un API file aperto per aggiornare una API definizione esistente

È possibile importare API definizioni solo per aggiornare una esistenteAPI, senza modificarne la configurazione dell'endpoint, nonché gli stadi e le variabili di fase o i riferimenti alle API chiavi.

L' import-to-update operazione può avvenire in due modalità: unione o sovrascrittura.

Quando un API (`A`) viene unito a un altro (`B`), il risultato API mantiene le definizioni di entrambi `A` e `B` se i due APIs non condividono definizioni in conflitto. Quando sorgono conflitti, le definizioni dei metodi di merging API (`A`) sovrascrivono le definizioni di metodo corrispondenti di merged (). API `B` Ad esempio, supponiamo che `B` abbia dichiarato i seguenti metodi per restituire le risposte `200` e `206`:

```
GET /a
POST /a
```

e `A` dichiari il metodo seguente per restituire le risposte `200` e `400`:

```
GET /a
```

Quando `A` viene unito a, il risultato produce `B` i seguenti API metodi:

```
GET /a
```

che restituisce le risposte `200` e `400` e

```
POST /a
```

che restituisce le risposte `200` e `206`.

L'unione di un API è utile quando avete scomposto le API definizioni esterne in più parti più piccole e desiderate applicare le modifiche solo da una di quelle parti alla volta. Ad esempio, ciò potrebbe verificarsi se più team sono responsabili di diverse parti di un progetto API e le modifiche sono disponibili a ritmi diversi. In questa modalità, gli elementi dell'esistente API che non sono definiti in modo specifico nella definizione importata vengono lasciati soli.

Quando un API (A) sovrascrive un altro API (B), il risultato API prende le definizioni della sovrascrittura API ()A. La sovrascrittura di un API è utile quando una API definizione esterna contiene la definizione completa di un. API In questa modalità, gli elementi di un esistente API che non sono definiti in modo specifico nella definizione importata vengono eliminati.

Per unire unAPI, invia una PUT richiesta a`https://apigateway.<region>.amazonaws.com/restapis/<restapi_id>?mode=merge`. Il valore del parametro `restapi_id` path specifica API a quale verrà unita la API definizione fornita.

Il seguente frammento di codice mostra un esempio di PUT richiesta di unione di una OpenAPI API definizioneJSON, come payload, con quella già specificata in Gateway. API API

```
PUT /restapis/<restapi_id>?mode=merge
Host:apigateway.<region>.amazonaws.com
Content-Type: application/json
Content-Length: ...

An OpenAPI API definition in JSON
```

L'operazione di fusione di aggiornamento richiede due API definizioni complete e le unisce. Per una modifica piccola e incrementale, puoi utilizzare l'operazione di aggiornamento risorsa.

Per sovrascrivere unAPI, invia una richiesta aPUT. `https://apigateway.<region>.amazonaws.com/restapis/<restapi_id>?mode=overwrite` Il parametro `restapi_id` path specifica API che verrà sovrascritto con le definizioni fornite. API

Il seguente frammento di codice mostra un esempio di richiesta di sovrascrittura con il payload di una definizione in formato: JSON OpenAPI

```
PUT /restapis/<restapi_id>?mode=overwrite
Host:apigateway.<region>.amazonaws.com
Content-Type: application/json
Content-Length: ...

An OpenAPI API definition in JSON
```

Se il parametro di query `mode` non viene specificato, si procede con l'unione.

> **Note**
>
> Le operazioni PUT sono idempotenti, ma non atomiche. Ciò significa che se un errore di sistema si verifica durante l'elaborazione, API può finire in uno stato negativo. Tuttavia, se si ripete correttamente l'operazione, la API si passa allo stesso stato finale come se la prima operazione fosse riuscita.

Impostare la proprietà Open API **`basePath`**

In Open API 2.0, è possibile utilizzare la `basePath` proprietà per fornire una o più parti del percorso che precedono ogni percorso definito nella `paths` proprietà. Poiché API Gateway offre diversi modi per esprimere il percorso di una risorsa, la API funzione Importa fornisce le seguenti opzioni per interpretare la `basePath` proprietà durante l'importazione: ignora, anteponi e dividi.

In OpenAPI 3.0 `basePath` non è più una proprietà di primo livello. APIGateway utilizza invece una variabile server come convenzione. La API funzione di importazione offre le stesse opzioni per interpretare il percorso di base durante l'importazione. Il percorso di base può essere identificato in questo modo:

- Se API non contiene alcuna `basePath` variabile, la API funzione di importazione controlla la `server.url` stringa per vedere se contiene un percorso successivo. `"/"` Se sì, il percorso viene usato come percorso di base.
- Se API contiene una sola `basePath` variabile, la API funzione di importazione la utilizza come percorso di base, anche se non è referenziata `server.url` in.
- Se API contiene più `basePath` variabili, la API funzione di importazione utilizza solo la prima come percorso di base.

Ignorare

Se il API file Apri ha un `basePath` valore di `/a/b/c` e la `paths` proprietà contiene `/e` e `/f`, quanto segue `POST` o `PUT` richiede:

```
POST /restapis?mode=import&basepath=ignore
```

```
PUT /restapis/api_id?basepath=ignore
```

produce le seguenti risorse inAPI:

- `/`
- `/e`
- `/f`

L'effetto è quello di trattarlo `basePath` come se non fosse presente e tutte le API risorse dichiarate vengono servite rispetto all'host. Questo può essere utilizzato, ad esempio, quando si dispone di un nome di dominio personalizzato con una API mappatura che non include un Base Path e un valore Stage che si riferisce alla fase di produzione.

> **Note**
>
> APIGateway crea automaticamente una risorsa root per voi, anche se non è dichiarata esplicitamente nel file di definizione.

In assenza di specifiche, `basePath` seleziona `ignore` per impostazione predefinita.

Metti come prefisso

Se nel file OpenAPI il valore di `basePath` è `/a/b/c` e la proprietà `paths` contiene `/e` e `/f`, la richiesta `POST` o `PUT` seguente:

```
POST /restapis?mode=import&basepath=prepend
```

```
PUT /restapis/api_id?basepath=prepend
```

produce le seguenti risorse in: API

- `/`
- `/a`
- `/a/b`
- `/a/b/c`
- `/a/b/c/e`
- `/a/b/c/f`

L'effetto è trattare il `basePath` come se specificasse risorse aggiuntive (senza metodo) e aggiungerle al set di risorse dichiarato. Questo può essere utilizzato, ad esempio, quando team diversi sono responsabili di diverse parti di un team API e `basePath` possono fare riferimento alla posizione del percorso per ogni API parte del team.

> **Note**
>
> APIGateway crea automaticamente risorse intermedie per te, anche se non sono dichiarate esplicitamente nella tua definizione.

Split

Se nel file OpenAPI il valore di `basePath` è `/a/b/c` e la proprietà `paths` contiene `/e` e `/f`, la richiesta `POST` o `PUT` seguente:

```
POST /restapis?mode=import&basepath=split
```

```
PUT /restapis/api_id?basepath=split
```

si traduce nelle seguenti risorse in: API

- `/`
- `/b`
- `/b/c`
- `/b/c/e`

- `/b/c/f`

L'effetto è quello di considerare la parte più in alto del percorso come l'inizio del percorso di ogni risorsa e di creare risorse aggiuntive (senza metodo) all'interno del percorso API stesso. `/a` Questo potrebbe, ad esempio, essere usato quando a è un nome d'arte che vuoi esporre come parte del tuo. API

AWS variabili per Open import API

È possibile utilizzare le seguenti AWS variabili nelle API definizioni aperte. APIGateway risolve le variabili quando viene importataAPI. Per specificare una variabile, utilizzare `${variable-name}`. La tabella seguente descrive le variabili disponibili. AWS

Nome della variabile	Descrizione
`AWS::AccountId`	L'ID AWS dell'account che importa ilAPI. Ad esempio, 123456789012.
`AWS::Partition`	La AWS partizione in cui viene importato. API Per le AWS regioni standard, la partizione è. `aws`
`AWS::Region`	La AWS regione in cui API viene importato. Ad esempio, `us-east-2` .

AWS esempio di variabili

L'esempio seguente utilizza AWS le variabili per specificare una AWS Lambda funzione per un'integrazione.

OpenAPI 3.0

```
openapi: "3.0.1"
info:
  title: "tasks-api"
  version: "v1.0"
```

```
paths:
  /:
    get:
      summary: List tasks
      description: Returns a list of tasks
      responses:
        200:
          description: "OK"
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: "#/components/schemas/Task"
        500:
          description: "Internal Server Error"
          content: {}
      x-amazon-apigateway-integration:
        uri:
          arn:${AWS::Partition}:apigateway:${AWS::Region}:lambda:path/2015-03-31/
functions/arn:${AWS::Partition}:lambda:${AWS::Region}:
${AWS::AccountId}:function:LambdaFunctionName/invocations
        responses:
          default:
            statusCode: "200"
        passthroughBehavior: "when_no_match"
        httpMethod: "POST"
        contentHandling: "CONVERT_TO_TEXT"
        type: "aws_proxy"
components:
  schemas:
    Task:
      type: object
      properties:
        id:
          type: integer
        name:
          type: string
        description:
          type: string
```

Errori e avvisi derivanti dall'importazione in Gateway API API

Quando si importa il file di definizione esterno in API Gateway, API Gateway potrebbe generare avvisi ed errori. Nelle sezioni seguenti vengono descritti gli errori e gli avvisi che potrebbero verificarsi durante l'importazione.

Errori durante l'importazione

Nel corso dell'importazione, possono essere generati errori per problematiche rilevanti come nel caso di un documento OpenAPI non valido. Gli errori vengono restituiti come eccezioni (ad esempio, `BadRequestException`) in una risposta non andata a buon fine. Quando si verifica un errore, la nuova API definizione viene eliminata e non viene apportata alcuna modifica a quella esistente. API

Avvisi durante l'importazione

Nel corso dell'importazione, gli avvisi possono essere generati per problematiche minori come nel caso di un riferimento mancante a un modello. Se viene visualizzato un avviso, l'operazione continuerà se l'espressione di `failonwarnings=false` interrogazione viene aggiunta alla richiesta. URL In caso contrario, gli aggiornamenti verranno ripristinati. Per impostazione predefinita, `failonwarnings` è impostato su `false`. In questi casi, gli avvisi vengono restituiti come campo nella risorsa risultante RestApi. In caso contrario, gli avvisi vengono restituiti come messaggio nell'eccezione.

Esporta un file REST API da API Gateway

Dopo aver creato e configurato un REST API in API Gateway, utilizzando la console API Gateway o in altro modo, puoi esportarlo in un API file aperto utilizzando API Gateway ExportAPI, che fa parte di Amazon API Gateway Control Service. Per utilizzare API Gateway ExportAPI, devi firmare API le tue richieste. Per ulteriori informazioni sulla firma delle richieste, consulta la AWS APIsezione Richieste di firma nella Guida IAM per l'utente. È possibile includere le estensioni di integrazione API Gateway, nonché le estensioni Postman, nel file Open API definition esportato.

> **Note**
>
> Quando esportate API utilizzando il AWS CLI, assicuratevi di includere il parametro extensions come mostrato nell'esempio seguente, per assicurarvi che l'`x-amazon-apigateway-request-validator`estensione sia inclusa:

```
aws apigateway get-export --parameters extensions='apigateway' --rest-api-id
 abcdefg123 --stage-name dev --export-type swagger latestswagger2.json
```

Non è possibile esportare un API file se i relativi payload non sono del `application/json` tipo. Se ci provi, riceverai una risposta di errore che indica che i modelli JSON corporei non sono stati trovati.

Richiesta di esportazione di un REST API

Con l'opzione ExportAPI, si esporta un REST API file esistente inviando una GET richiesta, specificandolo to-be-exported API come parte dei URL percorsi. La richiesta ha URL il seguente formato:

OpenAPI 3.0

```
https://<host>/restapis/<restapi_id>/stages/<stage_name>/exports/oas30
```

OpenAPI 2.0

```
https://<host>/restapis/<restapi_id>/stages/<stage_name>/exports/swagger
```

È possibile aggiungere la stringa di `extensions` query per specificare se includere le estensioni API Gateway (con il `integration` valore) o le estensioni Postman (con il `postman` valore).

Inoltre, è possibile impostare l'`Accept`intestazione su `application/json` o `application/yaml` ricevere l'output della API definizione rispettivamente in JSON o in YAML formato.

Per ulteriori informazioni sull'invio di GET richieste utilizzando API Gateway ExportAPI, vedere. GetExport

> **Note**
>
> Se definisci i modelli nel tuoAPI, per consentire a API Gateway di esportare il modello devono corrispondere al tipo di contenuto «application/json». In caso contrario, API Gateway genera un'eccezione con il messaggio di errore «Only found JSON non-body models for...».

I modelli devono contenere proprietà o essere definiti come un JSONSchema tipo particolare.

Scarica REST API Open API definition in JSON

Per esportare e scaricare un file REST API in API Definizioni aperte in JSON formato:

OpenAPI 3.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/oas30
Host: apigateway.<region>.amazonaws.com
Accept: application/json
```

OpenAPI 2.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/swagger
Host: apigateway.<region>.amazonaws.com
Accept: application/json
```

Qui *<region>* può essere, ad esempio, `us-east-1`. Per tutte le regioni in cui API Gateway è disponibile, consulta Regioni ed endpoint.

Scarica REST API Open API definition in YAML

Per esportare e scaricare un file REST API in API Definizioni aperte in YAML formato:

OpenAPI 3.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/oas30
Host: apigateway.<region>.amazonaws.com
Accept: application/yaml
```

OpenAPI 2.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/swagger
Host: apigateway.<region>.amazonaws.com
Accept: application/yaml
```

Scarica REST API Open API definition con le estensioni Postman in JSON

Per esportare e scaricare un file REST API in API Definizioni aperte con Postman in JSON formato:

OpenAPI 3.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/oas30?extensions=postman
Host: apigateway.<region>.amazonaws.com
Accept: application/json
```

OpenAPI 2.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/swagger?extensions=postman
Host: apigateway.<region>.amazonaws.com
Accept: application/json
```

Scarica REST API Open API definition con integrazione API Gateway in YAML

Per esportare e scaricare un file REST API in API definizioni aperte con integrazione API Gateway in YAML formato:

OpenAPI 3.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/oas30?extensions=integrations
Host: apigateway.<region>.amazonaws.com
Accept: application/yaml
```

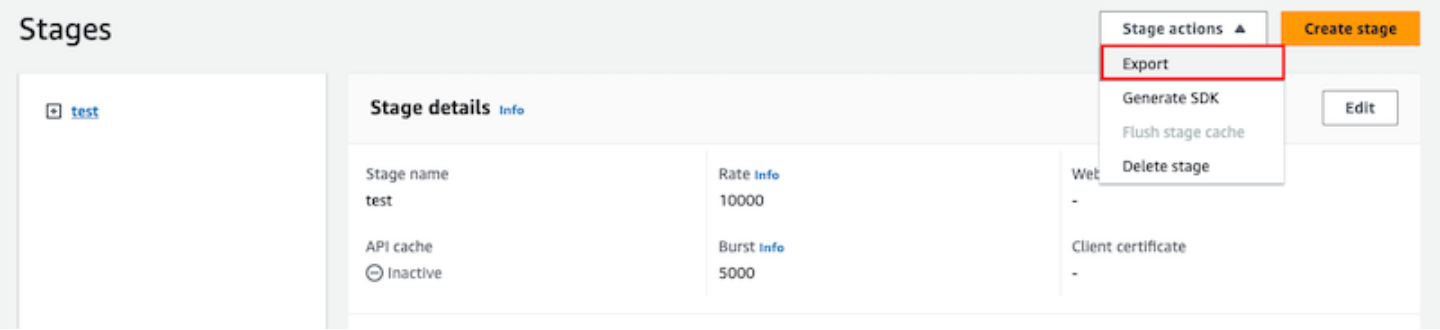
OpenAPI 2.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/swagger?
extensions=integrations
Host: apigateway.<region>.amazonaws.com
Accept: application/yaml
```

Esportazione REST API tramite la console API Gateway

Dopo la REST API distribuzione su uno stage, potete procedere all'esportazione dello stage API in un API file Open utilizzando la console API Gateway.

Nel riquadro Stages della console API Gateway, scegliete Stage actions, Export.



Specificate il tipo di API specifica, il formato e le estensioni per scaricare API la vostra API definizione aperta.

Pubblica API REST che i clienti possono richiamare

La semplice creazione e sviluppo di un'API di API Gateway non la rende automaticamente richiamabile dagli utenti. Per renderla richiamabile, è necessario distribuire l'API in una fase. Inoltre, potrebbe essere necessario personalizzare l'URL che verrà utilizzato dagli utenti per accedere all'API.

Puoi assegnarli un dominio che sia coerente con il tuo marchio o che sia più facile da ricordare dell'URL predefinito dell'API.

In questa sezione viene descritto come distribuire l'API e personalizzare l'URL fornito agli utenti per accedervi.

> **Note**
>
> Per aumentare la sicurezza delle API API Gateway, il dominio `execute-api.{region}.amazonaws.com`è registrato nella Public Suffix List (PSL). Per una maggiore sicurezza, consigliamo di utilizzare i cookie con un prefisso `__Host-` se hai bisogno di impostare cookie sensibili nel nome di dominio predefinito per le API API Gateway. Questa pratica ti aiuterà a difendere il tuo dominio dai tentativi CSRF (cross-site request forgery). Per ulteriori informazioni, consulta la pagina Impostazione cookie nella pagina Mozilla Developer Network.

Argomenti

- Implementazione REST APIs in Gateway API
- Nome di dominio personalizzato per REST APIs in API Gateway

Implementazione REST APIs in Gateway API

Dopo aver creato il tuoAPI, devi distribuirlo per renderlo richiamabile dai tuoi utenti.

Per distribuire unAPI, è necessario creare una API distribuzione e associarla a una fase. Una fase è un riferimento logico a uno stato del ciclo di vita dell'utente API (ad esempio,,, `devprod`,`beta`). `v2` APIle fasi sono identificate dall'APIID e dal nome della fase. Sono inclusi in URL quello che usi per richiamare ilAPI. Ogni fase è un riferimento denominato a una distribuzione di API ed è resa disponibile per la chiamata delle applicazioni client.

> **Important**
>
> Ogni volta che si aggiorna unaAPI, è necessario ridistribuirla API in una fase esistente o in una nuova fase. L'aggiornamento di un API include la modifica di percorsi, metodi, integrazioni, autorizzazioni, politiche delle risorse e qualsiasi altra cosa diversa dalle impostazioni dello stage.

Man mano che API si evolve, è possibile continuare a distribuirlo in fasi diverse come versioni diverse di. API Puoi anche distribuire i tuoi API aggiornamenti come distribuzione Canary Release. Ciò consente API ai vostri clienti di accedere, nella stessa fase, alla versione di produzione tramite la versione di produzione e alla versione aggiornata tramite la versione canary.

Per chiamare un deployedAPI, il client invia una richiesta contro un utente. API URL URLÈ determinato dal protocollo ((S) o HTTP (WSS)), dal nome host, dal nome della fase e dal percorso della risorsa (for RESTAPIs). API Il nome host e il nome dello stage determinano API la base. URL

Utilizzando il nome API di dominio predefinito, la base URL di a REST API (ad esempio) in una determinata fase (*`{stageName}`*) ha il seguente formato:

```
https://{restapi-id}.execute-api.{region}.amazonaws.com/{stageName}
```

Per rendere API la base predefinita URL più intuitiva, puoi creare un nome di dominio personalizzato (ad esempio,`api.example.com`) per sostituire il nome host predefinito di. API Per supportare più utenti APIs con il nome di dominio personalizzato, è necessario mappare uno API stage su un percorso di base.

Con un nome di dominio personalizzato *`{api.example.com}`* e lo API stage mappato su un percorso base (*`{basePath}`*) sotto il nome di dominio personalizzato, la base URL di a REST API diventa la seguente:

```
https://{api.example.com}/{basePath}
```

Per ogni fase, puoi ottimizzare le API prestazioni regolando i limiti predefiniti di limitazione delle richieste a livello di account e abilitando la memorizzazione nella cache. API Puoi anche abilitare la registrazione delle API chiamate verso CloudTrail or CloudWatch e selezionare un certificato client per il backend per autenticare le richieste. API Inoltre, è possibile sovrascrivere le impostazioni a livello di fase per i singoli metodi e definire variabili di fase per passare contesti di ambiente specifici dello stadio all'integrazione in fase di esecuzione. API

Le fasi consentono un solido controllo della versione di. API Ad esempio, puoi API implementare an su uno `test` stage e uno `prod` stage e utilizzare lo `test` stage come build di test e utilizzare lo `prod` stage come build stabile. Dopo che gli aggiornamenti hanno superato il test, puoi promuovere la fase `test` alla fase `prod`. La promozione può essere effettuata ridistribuendo lo stage API to the `prod` stage o aggiornando il valore di una variabile di stage dal nome della fase `test` a quello di. `prod`

In questa sezione, discutiamo come implementare un API utilizzando la console API Gateway o chiamando il Gateway. API REST API Per utilizzare altri strumenti, consulta la documentazione di AWS CLIo un AWS SDK.

Argomenti

- Creare una distribuzione per un API gateway REST API in
- Prepara un palco per un REST API in API Gateway
- Configurare una distribuzione di API Gateway Canary Release
- Gli aggiornamenti REST APIs che richiedono una ridistribuzione

Creare una distribuzione per un API gateway REST API in

In API Gateway, una REST API distribuzione è rappresentata da una risorsa Deployment. È simile a un eseguibile di un API che è rappresentato da una RestApirisorsa.

Affinché il client possa chiamare il tuoAPI, devi creare una distribuzione e associarvi una fase. Una fase è rappresentata da una risorsa Fase. Rappresenta un'istantanea diAPI, inclusi metodi, integrazioni, modelli, modelli di mappatura e autorizzatori Lambda (precedentemente noti come autorizzatori personalizzati). Quando si aggiorna ilAPI, è possibile ridistribuirlo associando una nuova fase alla API distribuzione esistente. La procedura di creazione di una fase è illustrata in the section called “Configurare una fase”.

Argomenti

- Crea distribuzione
- I passaggi successivi per la distribuzione API

Crea distribuzione

Le procedure seguenti mostrano come creare una distribuzione per un. REST API

AWS Management Console

È necessario averne creato uno REST API prima di distribuirlo per la prima volta. Per ulteriori informazioni, consulta Sviluppa REST APIs in API Gateway.

La console API Gateway consente di implementare an API creando una distribuzione e associandola a una fase nuova o esistente.

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel riquadro di APIsnavigazione, scegli API quello che desideri distribuire.
3. Nel riquadro Risorse, scegli Distribuisci API.
4. In Fase, procedi come segue:
   a. Per creare una nuova fase, seleziona Nuova fase, quindi immetti un nome in Nome fase. Facoltativamente, puoi immettere una descrizione dell'implementazione in Descrizione distribuzione.
   b. Per scegliere una fase esistente, seleziona il nome della fase nel menu a discesa. Se lo desideri, puoi specificare una descrizione della nuova implementazione in Descrizione distribuzione.
   c. Per creare un'implementazione non associata a una fase, seleziona Nessuna fase. Successivamente, puoi associare questa implementazione a una fase.
5. Seleziona Deploy (Implementa).

AWS CLI

Quando si crea una distribuzione, viene creata un'istanza della risorsa Distribuzione. È possibile utilizzare la console API Gateway, il AWS CLI AWS SDK, an o il API Gateway REST API per creare una distribuzione.

Per utilizzare il CLI per creare una distribuzione, usa il comando create-deployment:

```
aws apigateway create-deployment --rest-api-id <rest-api-id> --region <region>
```

non API è richiamabile finché non si associa questa distribuzione a una fase. Con una fase esistente è possibile eseguire questa operazione aggiornando la proprietà `deploymentId` della fase con l'ID di distribuzione appena creato (`<deployment-id>`).

```
aws apigateway update-stage --region <region> \
   --rest-api-id <rest-api-id> \
   --stage-name <stage-name> \
   --patch-operations op='replace',path='/deploymentId',value='<deployment-id>'
```

Quando si distribuisce una per API la prima volta, è possibile combinare la creazione della fase e la creazione della distribuzione contemporaneamente:

```
aws apigateway create-deployment --region <region> \
   --rest-api-id <rest-api-id> \
   --stage-name <stage-name>
```

Questo è ciò che viene fatto dietro le quinte nella console API Gateway quando si distribuisce una console per API la prima volta o quando la si ridistribuisce in una nuova faseAPI.

Per ridistribuire unAPI, esegui la stessa procedura. Puoi riutilizzare la stessa fase tutte le volte che si desideri.

I passaggi successivi per la distribuzione API

Di seguito sono riportati i passaggi successivi per la API distribuzione.

Modifica le impostazioni dello stage

Dopo aver distribuito uno, API è possibile modificare le impostazioni dello stage per abilitare o disabilitare la API cache, la registrazione o la limitazione delle richieste. Puoi anche scegliere un certificato client per il backend per autenticare API Gateway e impostare le variabili di fase per passare il contesto di implementazione all'integrazione in fase di esecuzione. API Per ulteriori informazioni, consulta Modifica le impostazioni della fase

Dopo aver modificato le impostazioni della fase, è necessario ridistribuire le API affinché le modifiche abbiano effetto.

> **Note**
>
> Se le impostazioni aggiornate, ad esempio l'abilitazione della registrazione, richiedono un nuovo IAM ruolo, è possibile aggiungere il IAM ruolo richiesto senza ridistribuire il. API Tuttavia, potrebbero essere necessari alcuni minuti prima che il nuovo IAM ruolo abbia effetto. Prima che ciò accada, le tracce delle API chiamate non vengono registrate anche se è stata abilitata l'opzione di registrazione.

Scegli diverse combinazioni in fase di implementazione

Poiché una distribuzione rappresenta un'APIistantanea e una fase definisce un percorso all'interno di un'istantanea, è possibile scegliere diverse combinazioni in fase di implementazione per controllare il modo in cui gli utenti richiamano le diverse versioni di. API Ciò è utile, ad

esempio, quando si desidera ripristinare API lo stato di una distribuzione precedente o unire un «ramo privato» di una distribuzione a quella pubblica. API

La procedura seguente mostra come eseguire questa operazione utilizzando lo Stage Editor nella console Gateway. API Si presume che sia necessario averlo distribuito API più di una volta.

1. Se non sei già nel riquadro Fasi, nel pannello di navigazione principale, scegli Fasi.
2. Seleziona la fase da aggiornare.
3. Nella scheda Cronologia delle distribuzioni seleziona l'implementazione da utilizzare per la fase.
4. Scegli Cambia implementazione attiva.
5. Conferma di voler cambiare l'implementazione attiva e scegli Cambia implementazione attiva nella finestra di dialogo Rendi attiva l'implementazione.

Passa i dati specifici della distribuzione al tuo. API

Per una distribuzione, puoi impostare o modificare le variabili di fase per passare i dati specifici della distribuzione all'integrazione in fase di esecuzione. API Puoi eseguire questa operazione nella scheda Stage Variables (Variabili di fase) in Stage Editor (Editor fasi). Per ulteriori informazioni, consulta le istruzioni in Usa le variabili di fase per un REST API in API Gateway.

Prepara un palco per un REST API in API Gateway

Una fase è un riferimento denominato a una distribuzione, che è un'istantanea diAPI. Utilizzare una Stage (Fase) per gestire e ottimizzare una specifica distribuzione. Ad esempio, puoi configurare le impostazioni di fase per abilitare la memorizzazione nella cache, personalizzare il throttling della richiesta, configurare la registrazione, definire le variabili di fase o collegare una release Canary a scopi di test. La sezione seguente mostra come creare e configurare lo stage.

Creazione di una nuova fase

Dopo la distribuzione iniziale, puoi aggiungere altre fasi e associarle alle distribuzioni esistenti. È possibile utilizzare la console API Gateway per creare una nuova fase oppure scegliere una fase esistente durante la distribuzione di unaAPI. In generale, è possibile aggiungere una nuova fase a una API distribuzione prima di ridistribuire il. API Per creare una nuova fase utilizzando la console API Gateway, procedi nel seguente modo:

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.

2. Scegli un. REST API
3. Nel riquadro di navigazione principale, scegli Fasi sotto unAPI.
4. Dal riquadro di navigazione Fasi scegli Crea fase.
5. Per Nome fase immetti un nome, ad esempio **`prod`**.

> **Note**
>
> I nomi di fasi possono contenere solo caratteri alfanumerici, trattini e caratteri di sottolineatura. La lunghezza massima è 128 caratteri.

6. (Facoltativo). In Descrizione inserisci una breve descrizione.
7. Per Distribuzione, seleziona la data e l'ora della API distribuzione esistente che desideri associare a questa fase.
8. In Impostazioni aggiuntive puoi specificare le impostazioni aggiuntive per la fase.
9. Scegli Crea fase.

Modifica le impostazioni della fase

Dopo una corretta implementazione di unAPI, lo stage viene popolato con le impostazioni predefinite. È possibile utilizzare la console o il API Gateway REST API per modificare le impostazioni dello stage, incluse la memorizzazione nella API cache e la registrazione. I passaggi seguenti mostrano come eseguire questa operazione utilizzando l'editor Stage della console API Gateway.

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli un. REST API
3. Nel riquadro di navigazione principale, scegli Fasi sotto unAPI.
4. Nel riquadro Stages (Fasi), selezionare il nome della fase.
5. Nella sezione Dettagli fase scegli Modifica.
6. (Facoltativo) In Descrizione fase modifica la descrizione.
7. In Impostazioni aggiuntive modifica le seguenti impostazioni:

   #### Impostazioni cache

   Per abilitare la API memorizzazione nella cache per lo stage, attiva Provision API cache. Quindi configura la memorizzazione nella cache a livello di metodo predefinita, la capacità

della cache, la crittografia dei dati della cache, Cache time-to-live (TTL) e tutti i requisiti per l'invalidazione della cache per chiave.

La memorizzazione nella cache non è attiva finché non attivi la cache a livello di metodo predefinita o non attivi la cache a livello di metodo per un metodo specifico.

Per ulteriori informazioni sulle impostazioni della cache, consulta Impostazioni della cache per REST APIs in API Gateway.

> **Note**
>
> Se abiliti la API memorizzazione nella cache per una API fase, il costo della memorizzazione nella cache potrebbe essere addebitato al tuo account AWS . API La memorizzazione nella cache non è idonea per il piano gratuito AWS .

Impostazioni di limitazione (della larghezza di banda della rete)

Per impostare obiettivi di limitazione a livello di fase per tutti i metodi associati, attiva Throttling. API

Per Rate(Tasso), inserire un tasso di destinazione. Questa è la velocità, espressa in richieste al secondo, con cui i token vengono aggiunti al bucket di token. La velocità a livello di fase non deve essere superiore alla velocità a livello di account come specificato in APIQuote del gateway per la configurazione e l'esecuzione di un REST API.

Per Burst (ottimizzazione), inserisci un tasso di destinazione. La frequenza di burst è la capacità del token bucket. Ciò consente di passare più richieste per un periodo di tempo rispetto al tasso di destinazione. Questo tasso di ottimizzazione a livello di fase non deve essere superiore al tasso di ottimizzazione a livello di account come specificato in APIQuote del gateway per la configurazione e l'esecuzione di un REST API.

> **Note**
>
> I tassi di limitazione (della larghezza di banda della rete) non sono limiti rigidi e vengono applicati sulla base del miglior tentativo. In alcuni casi, i client possono superare gli obiettivi impostati. Non fare affidamento sulla limitazione per controllare i

costi o bloccare l'accesso a un. API Valuta la possibilità Budget AWSdi utilizzarlo per monitorare i costi e AWS WAFgestire le API richieste.

Impostazioni di firewall e certificati

Per associare un AWS WAF Web ACL allo stage, selezionate un Web ACL dall'elenco a ACL discesa Web. Se lo desideri, scegli Blocca API richiesta se il Web ACL non può essere valutato (Fail-Close).

Per selezionare un certificato client per la fase, scegli un certificato dal menu a discesa Certificati client.

8. Scegli Continua.
9. Rivedi le modifiche e scegli Salva modifiche.
10. Per abilitare Amazon CloudWatch Logs per tutti i metodi associati a questa fase di questo API GatewayAPI, nella sezione Logs and tracing, scegli Modifica.

> **Note**
>
> Per abilitare CloudWatch i log, devi anche specificare il IAM ruolo che consente a API Gateway ARN di scrivere informazioni nei CloudWatch log per conto del tuo utente. A tale scopo, scegli Impostazioni dal pannello di navigazione APIsprincipale. Quindi, per il ruolo di CloudWatch registro, inserisci il ARN di un IAM ruolo.
> Per gli scenari applicativi più comuni, il IAM ruolo potrebbe allegare la policy gestita di`AmazonAPIGatewayPushToCloudWatchLogs`, che contiene la seguente dichiarazione sulla politica di accesso:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents",
                "logs:GetLogEvents",
```

```
                "logs:FilterLogEvents"
            ],
            "Resource": "*"
        }
    ]
}
```

Il IAM ruolo deve contenere anche la seguente dichiarazione di relazione di fiducia:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "apigateway.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Per ulteriori informazioni CloudWatch, consulta la Amazon CloudWatch User Guide.

11. Seleziona un livello di registrazione dal menu a discesa CloudWatch Logs. I livelli di registrazione sono i seguenti:
    - Disattivata: la registrazione non è attivata per questa fase.
    - Solo errori: la registrazione è abilitata solo per gli errori.
    - Errori e registri delle informazioni: la registrazione è abilitata per tutti gli eventi.
12. Seleziona Data tracing per fare in modo che API Gateway riferisca alla registrazione CloudWatch della traccia dei dati per la tua fase. Questo può essere utile per la risoluzione dei problemiAPIs, ma può comportare la registrazione di dati sensibili.

> **Note**
>
> Ti consigliamo di non utilizzare il tracciamento dei dati per la produzione. APIs

13. Seleziona Metriche dettagliate per fare in modo che API Gateway CloudWatch riferisca alle API metriche di`API calls`,, `LatencyIntegration latency`, `400 errors` e. `500 errors` Per ulteriori informazioni CloudWatch, consulta il monitoraggio di base e il monitoraggio dettagliato nella Amazon CloudWatch User Guide.

> **Important**
>
> Al tuo account viene addebitato l'accesso alle metriche a livello di metodo, ma non alle CloudWatch metriche a livello o a API livello di fase.

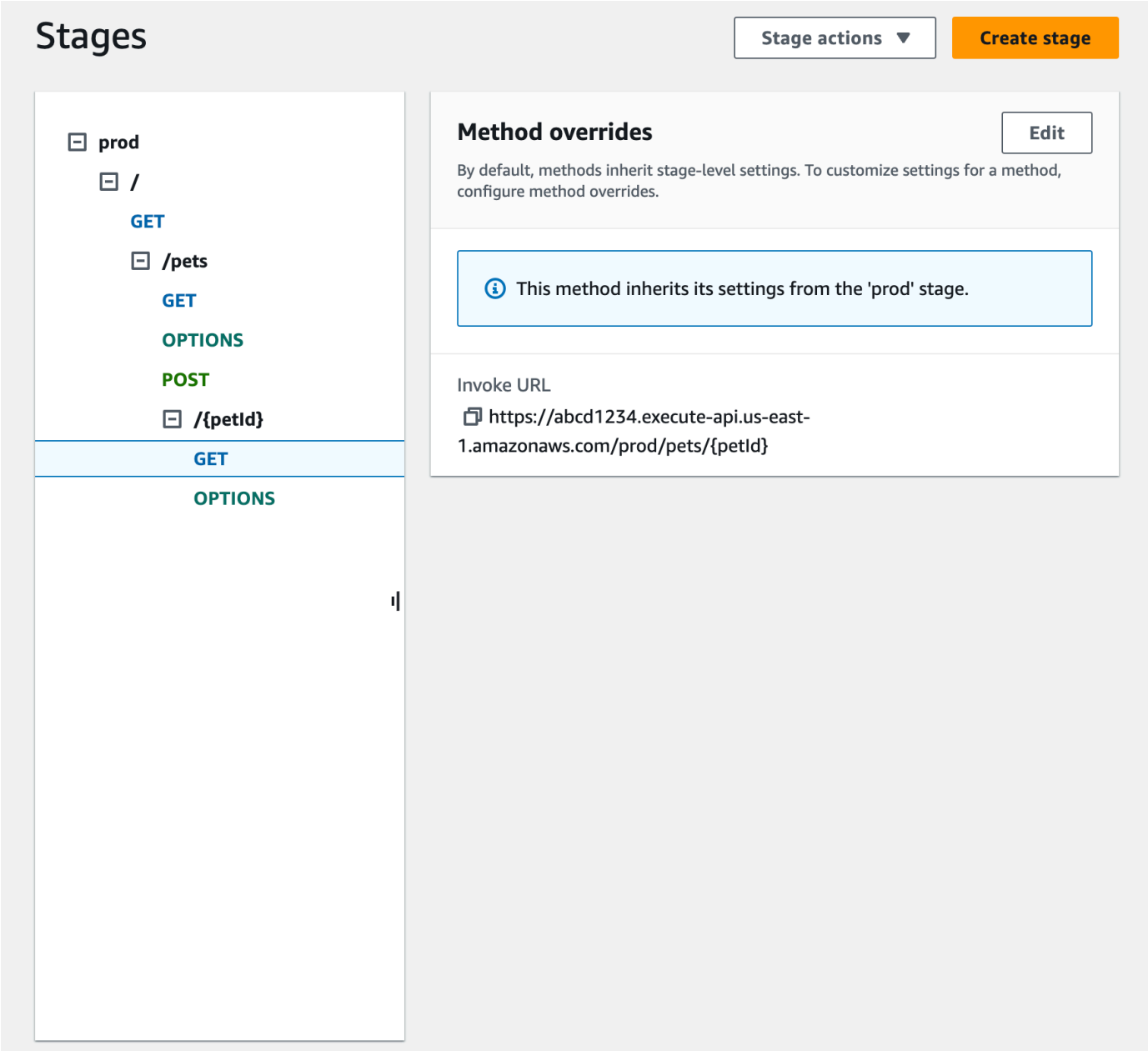
14. Per abilitare la registrazione degli accessi a una destinazione, attiva Registrazione accesso personalizzato.
15. Per la destinazione del registro di accesso ARN, immettete un gruppo ARN di log o uno stream Firehose.

    Il ARN formato per Firehose è. `arn:aws:firehose:{region}:{account-id}:deliverystream/amazon-apigateway-{your-stream-name}` Il nome dello stream Firehose deve essere. `amazon-apigateway-{your-stream-name}`
16. In Formato log immetti un formato di log. Per ulteriori informazioni sui formati di log di esempio, consulta the section called “CloudWatch formati di registro per API Gateway”.
17. Per abilitare il AWS X-Raytracciamento per lo API stage, selezionate Tracciamento X-Ray. Per ulteriori informazioni, consulta Traccia le richieste degli utenti sulle API REST utilizzando X-Ray in API Gateway.
18. Scegli Salva modifiche. Ridistribuite le vostre API per rendere effettive le nuove impostazioni.

Sostituzione delle impostazioni a livello di fase

Dopo aver personalizzato le impostazioni a livello di stage, potete sostituirle per ogni metodo. API Alcune di queste opzioni potrebbero comportare costi aggiuntivi per il tuo. Account AWS

1. Per configurare le sostituzioni dei metodi, espandi la fase nel pannello di navigazione secondario, quindi scegli un metodo.



2. Per Sostituzioni del metodo scegli Modifica.
3. Per attivare CloudWatch le impostazioni a livello di metodo, per CloudWatch Registri, seleziona un livello di registrazione.
4. Per attivare la registrazione della traccia dei dati per il tuo metodo, seleziona Tracciamento dei dati.

> **Note**
>
> Ti consigliamo di non utilizzare il tracciamento dei dati per la produzione. APIs

5. Per attivare i parametri dettagliati a livello di metodo, seleziona Parametri dettagliati. Al tuo account viene addebitato l'accesso alle metriche a livello di metodo, ma non alle CloudWatch metriche a livello di fase o a livello API di fase.
6. Per attivare la limitazione (della larghezza di banda della rete) a livello di metodo, seleziona Throttling. Immetti le opzioni appropriate a livello di metodo. Per ulteriori informazioni sulla limitazione, consulta the section called “Throttling”.
7. Per configurare la cache a livello di metodo, seleziona Abilita cache dei metodi. Se modificate l'impostazione predefinita della memorizzazione nella cache a livello di metodo nei dettagli dello stage, ciò non influisce su questa impostazione.
8. Seleziona Salva.

Eliminazione di una fase

Quando una fase non è più necessaria, puoi eliminarla per evitare di pagare per risorse inutilizzate. I passaggi seguenti mostrano come utilizzare la console API Gateway per eliminare una fase.

> **Warning**
>
> L'eliminazione di una fase potrebbe rendere inutilizzabile una parte o la totalità della corrispondente API da API parte dei chiamanti. L'eliminazione di una fase non può essere annullata, tuttavia è possibile creare nuovamente una fase e associarla alla stessa distribuzione.

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli un RESTAPI.
3. Nel riquadro di navigazione principale scegli Fasi.
4. Nel riquadro Fasi scegli la fase che vuoi eliminare, quindi seleziona Operazioni fase, Elimina fase.
5. Quando richiesto, immetti **`confirm`**, quindi scegli Elimina.

Configurare i tag per una fase API in API Gateway

In API Gateway è possibile aggiungere un tag a una fase API, rimuoverlo o visualizzarlo. A tale scopo, puoi utilizzare la console API Gateway, AWS CLI/SDK o l'API REST di API Gateway.

Una fase può anche ereditare i tag dalla sua API REST padre. Per ulteriori informazioni, consulta the section called “Eredità dei tag nell'API di Amazon API Gateway V1”.

Per ulteriori informazioni sull'assegnazione di tag alle risorse dell'API Gateway, consulta Applicazione di tag.

Argomenti

- Configurare i tag per una fase API utilizzando la console si API Gateway
- Configura i tag per una fase dell'API utilizzando AWS CLI
- Impostare i tag per una fase API utilizzando l'API REST di API Gateway

Configurare i tag per una fase API utilizzando la console si API Gateway

La procedura seguente descrive come configurare i tag per una fase API.

Per configurare i tag per una fase API mediante la console API Gateway

1. Accedere alla console API Gateway.
2. Seleziona un'API esistente o crea una nuova API che includa risorse, metodi e le integrazioni corrispondenti.
3. Seleziona una fase o distribuisci l'API in una nuova fase.
4. Nel riquadro di navigazione principale scegli Fasi.
5. Seleziona la scheda Tags (Tag). Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
6. Scegliere Gestisci tag.
7. In Editor di tag scegli Aggiungi nuovo tag. Immetti una chiave di tag (ad esempio, `Department`) nella colonna Key (Chiave), quindi immetti un valore di tag (ad esempio, `Sales`) nella colonna Value (Valore). Per salvare il tag scegli Salva.
8. Se necessario, ripeti la fase 5 per aggiungere altri tag alla fase API. Il numero massimo di tag per ogni fase è 50.
9. Per rimuovere un tag esistente dalla fase scegli Rimuovi.
10. Se l'API è stata implementata in precedenza nella console API Gateway, sarà necessario ridistribuirla per rendere effettive le modifiche.

Configura i tag per una fase dell'API utilizzando AWS CLI

È possibile impostare i tag per uno stadio API AWS CLI utilizzando il comando create-stage o il comando tag-resource. È possibile eliminare uno o più tag da una fase API utilizzando il comando untag-resource.

L'esempio seguente aggiunge un tag durante la creazione di uno stage: `test`

```
aws apigateway create-stage --rest-api-id abc1234 --stage-name test --description
 'Testing stage' --deployment-id efg456 --tag Department=Sales
```

L'esempio seguente aggiunge un tag a uno `prod` stage:

```
aws apigateway tag-resource --resource-arn arn:aws:apigateway:us-east-2::/
restapis/abc123/stages/prod --tags Department=Sales
```

L'esempio seguente rimuove il `Department=Sales` tag dallo `test` stage:

```
aws apigateway untag-resource --resource-arn arn:aws:apigateway:us-east-2::/
restapis/abc123/stages/test --tag-keys Department
```

Impostare i tag per una fase API utilizzando l'API REST di API Gateway

È possibile configurare i tag per una fase API utilizzando l'API REST di API Gateway con una delle operazioni seguenti:

- Richiamare `tags:tag` per aggiungere tag a una fase API.
- Richiamare `tags:untag` per eliminare uno o più tag da una fase API.
- Richiamare `stage:create` per aggiungere uno o più tag a una fase API in fase di creazione.

È anche possibile richiamare `tags:get` per descrivere i tag in una fase API.

Aggiunta di tag a una fase API

Dopo avere distribuito un'API (`m5zr3vnks7`) in una fase (`test`), è possibile aggiungere tag alla fase richiamando `tags:tag`. L'Amazon Resource Name (ARN) richiesto della fase (`arn:aws:apigateway:us-east-1::/restapis/m5zr3vnks7/stages/test`) deve avere codifica URL (`arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis%2Fm5zr3vnks7%2Fstages%2Ftest`).

```
PUT /tags/arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis%2Fm5zr3vnks7%2Fstages
%2Ftest

{
  "tags" : {
    "Department" : "Sales"
  }
}
```

Puoi anche usare la richiesta precedente per aggiornare un tag esistente in un nuovo valore.

È possibile aggiungere tag a una fase richiamando `stage:create` per la relativa creazione:

```
POST /restapis/<restapi_id>/stages

{
  "stageName" : "test",
  "deploymentId" : "adr134",
  "description" : "test deployment",
  "cacheClusterEnabled" : "true",
  "cacheClusterSize" : "500",
  "variables" : {
    "sv1" : "val1"
  },
  "documentationVersion" : "test",

  "tags" : {
    "Department" : "Sales",
    "Division" : "Retail"
  }
}
```

Rimozione di tag da una fase API

Per rimuovere il tag `Department` dalla fase, richiamare `tags:untag`:

```
DELETE /tags/arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis%2Fm5zr3vnks7%2Fstages
%2Ftest?tagKeys=Department
Host: apigateway.us-east-1.amazonaws.com
Authorization: ...
```

Per rimuovere più di un tag, usare un elenco di chiavi di tag separate da virgole nell'espressione di query, ad esempio `?tagKeys=Department,Division,…`.

Descrizione dei tag per una fase API

Per descrivere i tag esistenti per una fase specifica, richiamare `tags:get`:

```
GET /tags/arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis%2Fm5zr3vnks7%2Fstages
%2Ftags
Host: apigateway.us-east-1.amazonaws.com
Authorization: ...
```

La risposta con esito positivo è simile a quella riportata di seguito.

```
200 OK

{
    "_links": {
        "curies": {
            "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/
restapi-tags-{rel}.html",
            "name": "tags",
            "templated": true
        },
        "tags:tag": {
            "href": "/tags/arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis
%2Fm5zr3vnks7%2Fstages%2Ftags"
        },
        "tags:untag": {
            "href": "/tags/arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis
%2Fm5zr3vnks7%2Fstages%2Ftags{?tagKeys}",
            "templated": true
        }
    },
    "tags": {
        "Department": "Sales"
    }
}
```

Usa le variabili di fase per un REST API in API Gateway

Le variabili di fase sono coppie chiave-valore che puoi definire come attributi di configurazione associati a una fase di distribuzione di a. REST API Agiscono come variabili di ambiente e possono

essere utilizzate nei modelli di API configurazione e mappatura. Con le fasi di implementazione in API Gateway, puoi gestire più fasi di rilascio per ciascuna API e utilizzare le variabili di fase: puoi configurare una fase di API implementazione per interagire con diversi endpoint di backend.

Le variabili di fase non sono destinate ad essere utilizzate per i dati sensibili, come le credenziali. Per trasferire dati sensibili alle integrazioni, utilizza un autorizzatore. AWS Lambda È possibile passare dati sensibili alle integrazioni nell'output del provider di autorizzazioni Lambda. Per ulteriori informazioni, consulta the section called “Output da un autorizzatore API Gateway Lambda”.

Casi d'uso per le variabili di fase

Di seguito sono riportati i casi d'uso per le variabili di fase.

Specificare un endpoint di backend diverso

APIÈ possibile passare una `GET` richiesta come HTTP proxy all'host web di backend. Puoi utilizzare una variabile di fase in modo che quando API i chiamanti richiamano l'endpoint di produzione, Gateway chiami. API `example.com` Quindi, quando API i chiamanti richiamano la fase beta, API Gateway chiama un host web diverso, ad esempio. `beta.example.com` Allo stesso modo, le variabili di fase possono essere utilizzate per specificare un nome di AWS Lambda funzione diverso per ogni fase del sistema. API Non puoi utilizzare una variabile di fase per impostare un endpoint di integrazione diverso, ad esempio indirizzare la `GET` richiesta a un'integrazione HTTP proxy in una fase e a un'integrazione proxy Lambda in un'altra fase.

Quando si specifica un nome di funzione Lambda come valore della variabile di fase, è necessario configurare manualmente le autorizzazioni per la funzione Lambda. Quando si specifica una funzione Lambda nella console API Gateway, AWS CLI viene visualizzato un comando per configurare le autorizzazioni appropriate. A tale scopo è inoltre possibile utilizzare il seguente AWS CLI comando.

```
aws lambda add-permission --function-name "arn:aws:lambda:us-
east-2:123456789012:function:my-function" --source-arn "arn:aws:execute-
api:us-east-2:123456789012:api_id/*/HTTP_METHOD/resource" --principal
 apigateway.amazonaws.com --statement-id apigateway-access --action
 lambda:InvokeFunction
```

Passa le informazioni utilizzando modelli di mappatura

È possibile accedere alle variabili di fase nei modelli di mappatura o passare i parametri di configurazione al proprio AWS Lambda o HTTP al backend. Ad esempio, potresti voler riutilizzare

la stessa funzione Lambda per più fasi della API tua, ma la funzione dovrebbe leggere i dati da una tabella Amazon DynamoDB diversa a seconda della fase. Nei modelli di mappatura che generano la richiesta per la funzione Lambda è possibile usare le variabili di fase per passare il nome della tabella a Lambda.

Per utilizzare una variabile di fase, devi prima configurarla e poi assegnarle un valore. Ad esempio, per personalizzare l'endpoint di HTTP integrazione, crea prima la variabile `url` stage, quindi nella tua richiesta API di integrazione, inserisci il valore della variabile stage,. **`http://${stageVariables.url}`** Questo valore indica a API Gateway di sostituire la variabile stage `${}` in fase di esecuzione, a seconda della fase in esecuzioneAPI. Per ulteriori informazioni, consulta the section called “Imposta le variabili di fase per REST APIs in API Gateway”.

Imposta le variabili di fase per REST APIs in API Gateway

Questa sezione mostra come impostare diverse variabili di fase per due fasi di distribuzione di un campione API utilizzando la console Amazon API Gateway. Per capire come utilizzare le variabili di fase in API Gateway, ti consigliamo di seguire tutte le procedure in questa sezione.

Prerequisiti

Prima di iniziare, verifica che siano soddisfatti i seguenti requisiti preliminari:

- È necessario disporre di un file API disponibile in API Gateway. Segui le istruzioni in Sviluppa REST APIs in API Gateway.
- È necessario averlo distribuito API almeno una volta. Segui le istruzioni in Implementazione REST APIs in Gateway API.
- È necessario aver creato la prima fase per una distribuzioneAPI. Segui le istruzioni in Creazione di una nuova fase.

Richiama un HTTP endpoint tramite una variabile API di fase

Questa procedura descrive come creare una variabile di fase per un HTTP endpoint e due fasi per il tuo. API Inoltre, create le variabili di fase, `urlstageName`, e `function` che vengono utilizzate nelle seguenti procedure di questa sezione.

Per richiamare un HTTP endpoint tramite e API con una variabile di fase

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.

2. Crea un metodoAPI, quindi crea un `GET` metodo sulla API risorsa principale. Imposta il tipo di integrazione su HTTPe imposta l'endpoint URL su**`http://${stageVariables.url}`**.
3. Distribuiscili API in una nuova fase denominata. **`beta`**
4. Nel riquadro di navigazione scegli Fasi, quindi seleziona la fase beta.
5. Nella scheda Variabili di fase scegli Modifica.
6. Scegli Aggiungi variabile di fase.
7. Per Nome, immetti **`url`**. In Valore, inserisci **`httpbin.org/get`**.
8. Scegli Aggiungi variabile di fase, quindi effettua le seguenti operazioni:

   Per Nome, immetti **`stageName`**. In Valore, inserisci **`beta`**.
9. Scegli Aggiungi variabile di fase, quindi effettua le seguenti operazioni:

   Per Nome, immetti **`function`**. In Valore, inserisci **`HelloWorld`**.
10. Selezionare Salva.
11. Ora crea una seconda fase. Dal riquadro di navigazione Fasi scegli Crea fase. In Stage name (Nome fase) immettere **`prod`**. Seleziona un'implementazione recente da Implementazione e scegli Crea.
12. Come per la fase beta, impostate le stesse tre variabili di fase (url e function) su valori diversi (**`petstore-demo-endpoint.execute-api.com/petstore/petsprod`**, e**`HelloEveryone`**), rispettivamente. stageName
13. Nel riquadro di navigazione Stages (Fasi) scegli beta. In Dettagli dello stage, scegli l'icona di copia per copiare API l'invokeURL, quindi inserisci API l'invoke URL in un browser web. Ciò avvia la `GET` richiesta della fase beta sulla risorsa principale di. API

> **Note**
>
> Il URL collegamento Invoke punta alla risorsa principale di API nella sua fase beta. L'immissione di URL in un browser Web richiama il `GET` metodo della fase beta sulla risorsa root. Se i metodi sono definiti su risorse secondarie e non sulla risorsa root stessa, l'immissione di tali URL in un browser Web restituisce una risposta `{"message":"Missing Authentication Token"}` di errore. In questo caso, è necessario aggiungere il nome di una risorsa secondaria specifica al link Invoke URL.

14. La risposta che si ottiene dalla richiesta `GET` della fase beta è mostrata più avanti. Puoi verificare il risultato anche utilizzando un browser per accedere a http://httpbin.org/get. Questo valore è stato assegnato alla variabile `url` nella fase beta. Le due risposte sono identiche.
15. Nel riquadro di navigazione Stages (Fasi) scegli la fase prod. In Stage details, scegli l'icona di copia per copiare la tua API invokeURL, quindi inserisci la tua API invoke URL in un browser web. Questo avvia la `GET` richiesta prod stage sulla risorsa root di. API
16. La risposta che si ottiene dalla richiesta `GET` della fase prod è mostrata più avanti. È possibile verificare il risultato utilizzando un browser per accedere a http://petstore-demo-endpoint.execute-api.com/petstore/pets. Questo valore è stato assegnato alla variabile `url` nella fase prod. Le due risposte sono identiche.

Passa i HTTP metadati specifici dello stadio in un backend

Questa procedura descrive come utilizzare il valore di una variabile di fase in un'espressione di parametro di query per passare metadati specifici dello stadio in un backend. HTTP Useremo la variabile `stageName` stage dichiarata nella procedura precedente.

Per passare metadati specifici dello stadio in un backend HTTP

1. Nel riquadro di navigazione delle risorse, scegli il metodo. GET

   Per aggiungere un parametro della stringa di query al metodoURL, scegliete la scheda Richiesta metodo, quindi nella sezione Impostazioni della richiesta del metodo, scegliete Modifica.
2. Scegliete i parametri della stringa di URL query ed effettuate le seguenti operazioni:
   a. Scegliere Add query string (Aggiungi stringa di query).
   b. Per Nome, immetti **`stageName`**.
   c. Mantieni Obbligatorio e Caching disattivati.
3. Selezionare Salva.
4. Scegli la scheda Richiesta di integrazione, quindi seleziona Modifica nella sezione Impostazioni della richiesta di integrazione.
5. Per Endpoint URL, aggiungi **`?stageName=${stageVariables.stageName}`** al URL valore definito in precedenza, in modo che l'intero endpoint URL sia. **`http://${stageVariables.url}?stageName=${stageVariables.stageName}`**
6. Scegli Deploy API e seleziona la fase beta.

7. Nel riquadro di navigazione principale scegli Fasi. Nel riquadro di navigazione Stages (Fasi) scegli beta. In Dettagli dello stage, scegli l'icona di copia per copiare API la tua chiamataURL, quindi inserisci la API chiamata URL in un browser web.

> **Note**
>
> Qui utilizziamo la fase beta perché l'HTTPendpoint (come specificato dalla `url` variabile "http://httpbin.org/get «) accetta le espressioni dei parametri di query e le restituisce come `args` oggetto nella sua risposta.

8. Si ottiene la risposta seguente. `beta`, assegnato alla variabile di fase `stageName`, viene passata nel back-end come argomento `stageName`.

```
{
  "args": {
    "stageName": "beta"
  },
  "headers": {
    "Accept": "application/json",
    "Host": "httpbin.org",
    "User-Agent": "AmazonAPIGateway_abcd1234",
    "X-Amzn-Apigateway-Api-Id": "abcd1234",
    "X-Amzn-Trace-Id": "Self=1-abcd-11111111111111;Root=1-11111111-111111111111111"
  },
  "origin": "192.0.2.9",
  "url": "http://httpbin.org/get?stageName=beta"
}
```

Richiama una funzione Lambda tramite e con API una variabile di fase

Questa procedura descrive come utilizzare una variabile stage per chiamare una funzione Lambda come backend di. API Si utilizza la variabile `function` stage dichiarata in. Richiama un HTTP endpoint tramite una variabile API di fase

Quando imposti una funzione Lambda come valore di una variabile di fase, usa il nome locale della funzione, se possibile includendo l'alias o la specifica della versione, come in **`HelloWorld`**, **`HelloWorld:1`** o **`HelloWorld:alpha`**. Non utilizzate la funzione ARN (ad esempio,**`arn:aws:lambda:us-east-1:123456789012:function:HelloWorld`**). La console API Gateway assume il valore della variabile stage per una funzione Lambda come nome di funzione non qualificato ed espande la variabile di fase specificata in un. ARN

Per richiamare una funzione Lambda tramite e con API una variabile di fase

1. Crea una funzione Lambda denominata **`HelloWorld`** utilizzando il runtime Node.js predefinito. Il codice deve contenere quanto segue:

```
export const handler = function(event, context, callback) {
    if (event.stageName)
        callback(null, 'Hello, World! I\'m calling from the ' + event.stageName + '
 stage.');
    else
        callback(null, 'Hello, World! I\'m not sure where I\'m calling from...');
};
```

   Per ulteriori informazioni su come creare una funzione Lambda, consulta Guida introduttiva alla REST API console.

2. Nel riquadro Risorse seleziona Crea risorsa, quindi procedi come segue:

   a. Per Percorso risorsa, seleziona /.

   b. Per Resource Name (Nome risorsa) immetti **`lambdav1`**.

   c. Scegli Crea risorsa.

3. Scegli la risorsa /lambdav1, quindi scegli il metodo Create.

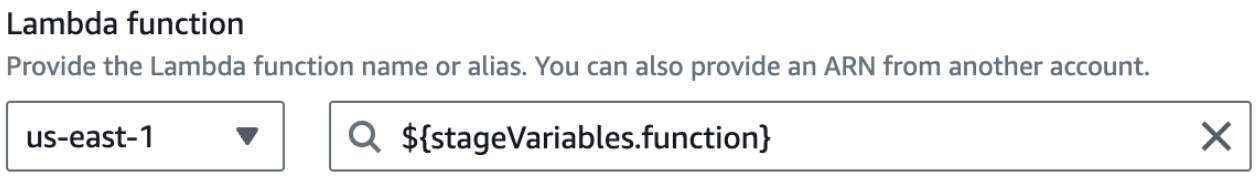
   Successivamente, esegui queste operazioni:

   a. In Tipo di metodo, seleziona GET.

   b. Per Tipo di integrazione seleziona Funzione Lambda.

   c. Mantieni l'opzione Integrazione proxy Lambda disattivata.

   d. Per Lambda function (Funzione Lambda), immetti `${stageVariables.function}`.

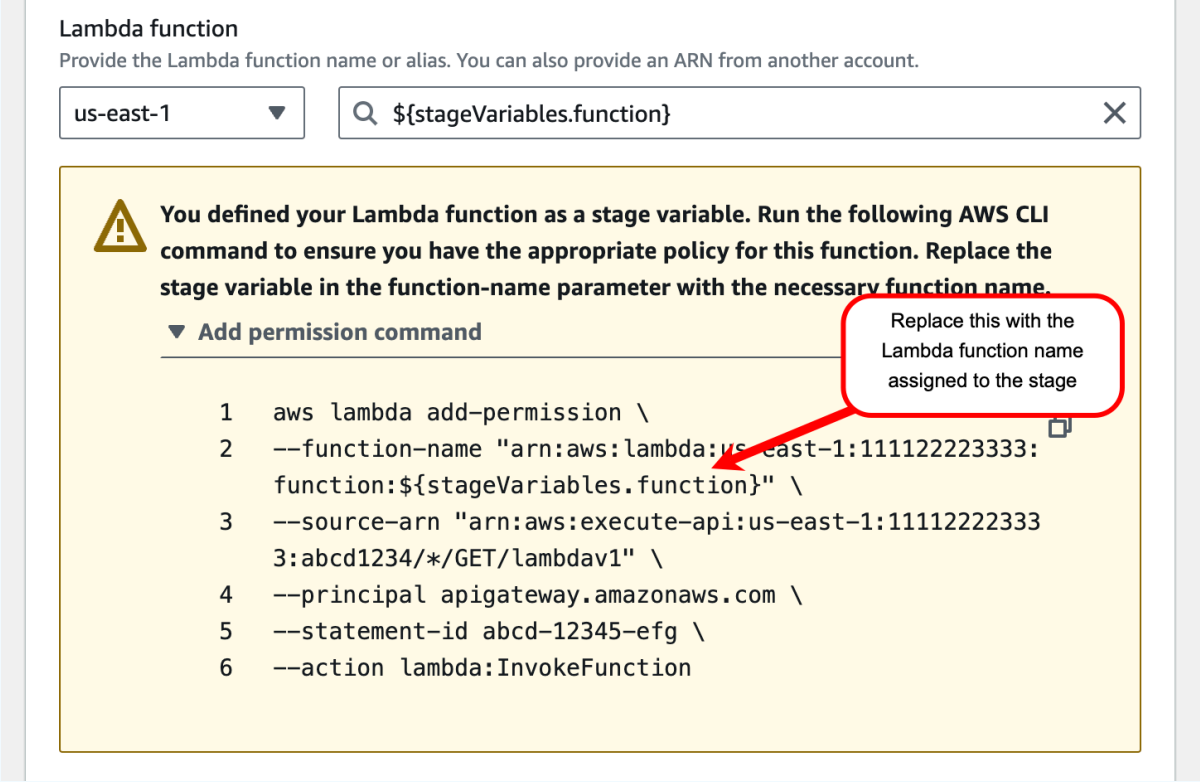


> **Tip**
>
> Quando richiesto con il comando Aggiungi autorizzazione, copia il comando. AWS CLI Esegui il comando per ciascuna funzione Lambda che sarà assegnata alla

variabile di fase `function`. Ad esempio, se il `$stageVariables.function` valore è`HelloWorld`, esegui il AWS CLI comando seguente:

```
aws lambda add-permission --function-name arn:aws:lambda:us-
east-1:account-id:function:HelloWorld --source-arn arn:aws:execute-
api:us-east-1:account-id:api-id/*/GET/lambdav1 --principal
 apigateway.amazonaws.com --statement-id statement-id-guid --action
 lambda:InvokeFunction
```

In caso contrario, riceverai una risposta `500 Internal Server Error` quando verrà invocato il metodo. Sostituisci `${stageVariables.function}` con il nome della funzione Lambda assegnato alla variabile di fase.



e. Scegli Crea metodo.

4. Distribuiscilo sia nella API fase prod che in quella beta.

5. Nel riquadro di navigazione principale scegli Fasi. Nel riquadro di navigazione Stages (Fasi) scegli beta. In Stage details, scegli l'icona di copia per copiare la tua API invokeURL, quindi inserisci la tua API invoke URL in un browser web. Aggiungi **`/lambdav1`** a URL prima di premere invio.

Si ottiene la risposta seguente.

```
"Hello, World! I'm not sure where I'm calling from..."
```

Passaggio dei metadati specifici delle fasi a una funzione Lambda mediante una variabile di fase

Questa procedura descrive come utilizzare una variabile di fase per passare i metadata di configurazione specifici delle fasi in una funzione Lambda. Crei un metodo `POST` e un modello di mappatura di input per generare il payload utilizzando la variabile di fase `stageName` dichiarata precedentemente.

Per passare metadati specifici dello stadio a una funzione Lambda tramite una variabile di fase

1. Scegliete la risorsa /lambdav1, quindi scegliete il metodo Create.

   Successivamente, esegui queste operazioni:

   a. In Tipo di metodo, seleziona POST.
   b. Per Tipo di integrazione seleziona Funzione Lambda.
   c. Mantieni l'opzione Integrazione proxy Lambda disattivata.
   d. Per Lambda function (Funzione Lambda), immetti `${stageVariables.function}`.
   e. Quando richiesto con il comando Aggiungi autorizzazione, copia il comando. AWS CLI Esegui il comando per ciascuna funzione Lambda che sarà assegnata alla variabile di fase `function`.
   f. Scegli Crea metodo.
2. Scegli la scheda Richiesta di integrazione, quindi seleziona Modifica nella sezione Impostazioni della richiesta di integrazione.
3. Scegli Modelli di mappatura, quindi seleziona Aggiungi modello di mappatura.
4. Per Tipo di contenuto inserisci **`application/json`**.
5. Per Corpo del modello inserisci il seguente modello:

```
#set($inputRoot = $input.path('$'))
{
    "stageName" : "$stageVariables.stageName"
}
```

> **Note**
>
> In un modello di mappatura il riferimento a una variabile di fase deve essere racchiuso tra apici (come in `"$stageVariables.stageName"` o `"${stageVariables.stageName}"`), mentre altrove non si devono utilizzare gli apici (come in `${stageVariables.function}`).

6. Selezionare Salva.
7. Distribuiscilo sia nella API fase beta che in quella di produzione.
8. Per utilizzare un REST API client per passare metadati specifici dello stadio, procedi come segue:

   a. Nel riquadro di navigazione Stages (Fasi) scegli beta. In Dettagli sullo stage, scegli l'icona di copia per copiare la tua API invokeURL, quindi inserisci la tua invoke URL nel campo API di immissione di un client. REST API Aggiungi **`/lambdav1`** prima di inviare la richiesta.

      Si ottiene la risposta seguente.

      ```
      "Hello, World! I'm calling from the beta stage."
      ```

   b. Nel riquadro di navigazione Fasi scegli prod. Nella sezione Dettagli dello stage, scegli l'icona di copia per copiare API la tua chiamataURL, quindi inserisci la tua chiamata URL nel campo API di immissione di un client. REST API Aggiungi **`/lambdav1`** prima di inviare la richiesta.

      Si ottiene la risposta seguente.

      ```
      "Hello, World! I'm calling from the prod stage."
      ```

9. Per utilizzare la funzionalità Test per trasmettere i metadati specifici della fase, procedi come segue:

   a. Nel riquadro di navigazione Risorse scegli la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
   b. Per function immetti **`HelloWorld`**.
   c. In stageName, immettere **`beta`**.
   d. Scegli Test (Esegui test). Non è necessario aggiungere un corpo alla richiesta `POST`.

      Si ottiene la risposta seguente.

```
"Hello, World! I'm calling from the beta stage."
```

e. Puoi ripetere i passaggi precedenti per testare la fase Prod. In stageName, immettere **`Prod`**.

   Si ottiene la risposta seguente.

```
"Hello, World! I'm calling from the prod stage."
```

APIRiferimento alle variabili di fase Gateway per REST APIs in Gateway API

È possibile utilizzare le variabili di fase API Gateway nei seguenti casi.

Espressioni di mappatura dei parametri

Una variabile di fase può essere utilizzata in un'espressione di mappatura dei parametri per il parametro di intestazione della richiesta o della risposta di un API metodo, senza alcuna sostituzione parziale. Nell'esempio che segue si fa riferimento alla variabile di fase senza il simbolo $ e il simbolo `{...}` di chiusura.

- `stageVariables.<variable_name>`

Modelli di mappatura

Una variabile di fase può essere utilizzata ovunque in un modello di mappatura, come mostrato negli esempi seguenti.

- `{ "name" : "$stageVariables.<variable_name>"}`
- `{ "name" : "${stageVariables.<variable_name>}"}`

HTTPintegrazione URIs

Una variabile di fase può essere utilizzata come parte di un'HTTPintegrazioneURL, come illustrato negli esempi seguenti:

- Un completo URI senza protocollo — `http://${stageVariables.<variable_name>}`
- Un dominio completo – `http://${stageVariables.<variable_name>}/resource/operation`

- Un sottodominio – `http://${stageVariables.<variable_name>}.example.com/resource/operation`
- Un percorso – `http://example.com/${stageVariables.<variable_name>}/bar`
- Una stringa di query – `http://example.com/foo?q=${stageVariables.<variable_name>}`

AWS integrazione URIs

Una variabile di fase può essere utilizzata come parte dei componenti dell' AWS URIazione o del percorso, come illustrato nell'esempio seguente.

- `arn:aws:apigateway:<region>:<service>:${stageVariables.<variable_name>}`

AWS integrazione URIs (funzioni Lambda)

Una variabile di fase può essere utilizzata al posto del nome o della versione/dell'alias di una funzione Lambda, come mostrato negli esempi seguenti.

- `arn:aws:apigateway:<region>:lambda:path/2015-03-31/functions/arn:aws:lambda:<region>:<account_id>:function:${stageVariables.<function_variable_name>}/invocations`
- `arn:aws:apigateway:<region>:lambda:path/2015-03-31/functions/arn:aws:lambda:<region>:<account_id>:function:<function_name>:${stageVariables.<version_variable_name>}/invocations`

> **Note**
>
> Per utilizzare una variabile stage per una funzione Lambda, la funzione deve trovarsi nello stesso account di. API Le variabili di fase non supportano le funzioni Lambda tra più account.

Bacino d'utenza di Amazon Cognito

Una variabile stage può essere utilizzata al posto di un pool di utenti Amazon Cognito per un `COGNITO_USER_POOLS` autorizzatore.

- `arn:aws:cognito-idp:<region>:<account_id>:userpool/${stageVariables.<variable_name>}`

AWS credenziali di integrazione

Una variabile di fase può essere utilizzata come parte della credenziale AWS utente/ruoloARN, come illustrato nell'esempio seguente.

- `arn:aws:iam::<account_id>:${stageVariables.<variable_name>}`

Configurare una distribuzione di API Gateway Canary Release

La versione Canary è una strategia di sviluppo software in cui una nuova versione di API (così come di altro software) viene distribuita a scopo di test e la versione base rimane utilizzata come versione di produzione per le normali operazioni nella stessa fase. In questa documentazione la versione di base viene detta release di produzione. È possibile applicare una release Canary a qualsiasi versione non di produzione a scopo di test.

In una distribuzione Canary Release, il API traffico totale viene suddiviso casualmente in una versione di produzione e una versione Canary con un rapporto preconfigurato. In genere, la versione Canary riceve una piccola percentuale di API traffico e la versione di produzione occupa il resto. Le API funzionalità aggiornate sono visibili solo al API traffico che attraversa le isole Canarie. Puoi modificare la percentuale di traffico nella release Canary per ottimizzare le prestazioni o la copertura dei test.

Mantenendo limitato il traffico nella release Canary e la selezione casuale, la maggior parte degli utenti non riscontra conseguenze negative a causa di bug potenziali nella nuova versione e nessun utente riscontra conseguenze negative durature.

Se i parametri di test soddisfano i requisiti, puoi promuovere la release Canary a release di produzione e disabilitarne la distribuzione. Le nuove caratteristiche vengono così rese disponibili nella fase di produzione.

Argomenti

- Distribuzione della versione Canary in Gateway API
- Creazione di una distribuzione di una release Canary
- Aggiornamento di una release Canary
- Promozione di una release Canary

- Disabilitazione di una release Canary

Distribuzione della versione Canary in Gateway API

In API Gateway, una versione canary release utilizza la fase di distribuzione per la versione di produzione della versione base di anAPI, e allega alla fase una versione canary per le nuove versioni, relativa alla versione base, di. API La fase è associata alla distribuzione iniziale e la release Canary alle distribuzioni successive. All'inizio, sia lo stage che il canary puntano alla stessa versione. API In questa sezione i termini fase e release di produzione vengono usati in modo intercambiabile, come pure i termini Canary e release Canary.

Per implementare una versione canary API con una release canary, è necessario creare una distribuzione canary release aggiungendo impostazioni canary alla fase di distribuzione regolare. Le impostazioni canary descrivono la versione canary sottostante e la fase rappresenta la versione di produzione di questa distribuzione. API Per aggiungere le impostazioni Canary, imposta `canarySettings` nella fase di distribuzione e specifica quanto segue:

- Un ID di distribuzione, inizialmente identico all'ID della distribuzione della versione di base impostato nella fase.
- Una percentuale di API traffico, compresa tra 0,0 e 100,0 inclusi, per la versione Canary.
- Variabili di fase per la release Canary che possono sovrascrivere le variabili di fase per la release di produzione.
- L'uso della cache stage per le richieste Canary, se useStageCacheè impostata e la API memorizzazione nella cache è abilitata sullo stage.

Dopo l'abilitazione di una release Canary, la fase di distribuzione non può essere associata a un'altra distribuzione di una release non Canary fino a quando la release Canary non viene disabilitata e le impostazioni Canary non vengono rimosse dalla fase.

Quando abiliti la registrazione API dell'esecuzione, la versione Canary ha i propri log e le proprie metriche generati per tutte le richieste Canary. Vengono segnalate a un gruppo di log Logs in fase di produzione e a un gruppo di log CloudWatch Logs specifico per Canary. CloudWatch Lo stesso vale per il logging degli accessi. I registri separati, specifici per Canary, sono utili per convalidare le nuove API modifiche e decidere se accettarle e promuovere l'immissione delle Canarie nella fase di produzione, oppure per eliminare le modifiche e annullare l'emissione per Canary dalla fase di produzione.

Il gruppo di log delle esecuzioni della fase di produzione è denominato `API-Gateway-Execution-Logs/{rest-api-id}/{stage-name}` e il gruppo di log delle esecuzioni della release Canary è denominato `API-Gateway-Execution-Logs/{rest-api-id}/{stage-name}/Canary`. Per il logging degli accessi, è necessario creare un nuovo gruppo di log o sceglierne uno esistente. Al nome del gruppo di log degli accessi della release Canary scelto viene aggiunto il suffisso `/Canary`.

Una versione canary può utilizzare lo stage cache, se abilitato, per memorizzare le risposte e utilizzare le voci memorizzate nella cache per restituire i risultati alle successive richieste Canary, entro un periodo preconfigurato (). time-to-live TTL

In una distribuzione Canary Release, la release di produzione e la release canary di API possono essere associate alla stessa versione o a versioni diverse. Quando sono associate a versioni diverse, le risposte per le richieste di produzione e Canary vengono memorizzate nella cache separatamente e la cache di fase restituisce i risultati corrispondenti per le richieste di produzione e Canary. Quando la release di produzione e la release Canary sono associate alla stessa distribuzione, la cache di fase usa una singola chiave cache per entrambi i tipi di richieste e restituisce la stessa risposta per le stesse richieste dalla release di produzione e dalla release Canary.

Creazione di una distribuzione di una release Canary

Si crea una distribuzione Canary Release quando si distribuiscono le impostazioni API with canary come input aggiuntivo all'operazione di creazione della distribuzione.

Puoi anche creare una distribuzione di una release Canary da una distribuzione non Canary esistente effettuando una richiesta `stage:update` per aggiungere le impostazioni Canary nella fase.

Quando crei una distribuzione di una release non Canary, puoi specificare un nome di fase non esistente. APIGateway ne crea uno se lo stadio specificato non esiste. Non puoi tuttavia specificare un nome di fase non esistente quando crei una distribuzione di una release Canary. Verrà visualizzato un errore e API Gateway non creerà alcuna distribuzione in versione canary.

È possibile creare una distribuzione Canary Release in API Gateway utilizzando la console API Gateway, il AWS CLI, o un. AWS SDK

Argomenti

- Crea una distribuzione Canary utilizzando la console Gateway API
- Crea un dispiegamento canarino usando il AWS CLI

Crea una distribuzione Canary utilizzando la console Gateway API

Per utilizzare la console API Gateway per creare una distribuzione Canary Release, segui le istruzioni seguenti:

Per creare la distribuzione di una release Canary iniziale

1. Accedi alla console API Gateway.
2. Scegline uno esistente REST API o creane uno nuovo RESTAPI.
3. Nel riquadro di navigazione principale, scegli Risorse, quindi scegli Distribuisci. API Segui le istruzioni sullo schermo in Deploy API per implementarlo in una nuova faseAPI.

   Finora, l'hai implementato in una fase di rilascio API di produzione. Successivamente, configurate le impostazioni di Canary sullo stage e, se necessario, abilitate anche la memorizzazione nella cache, impostate le variabili di fase o configurate i log di API esecuzione o di accesso.
4. Per abilitare la API memorizzazione nella cache o associare un AWS WAF Web ACL allo stage, nella sezione Dettagli dello stage, scegli Modifica. Per ulteriori informazioni, consulta the section called “Impostazioni cache” o the section called “Per associare un ACL AWS WAF Web a uno stadio API Gateway API utilizzando la console API Gateway”.
5. Per configurare la registrazione degli accessi o delle esecuzioni, scegli Modifica nella sezione Log e tracciamento e segui le istruzioni visualizzate sullo schermo. Per ulteriori informazioni, consulta Configurare la CloudWatch registrazione per REST APIs Gateway API.
6. Per impostare le variabili di fase, scegli la scheda Variabili di fase e segui le istruzioni visualizzate sullo schermo per aggiungere o modificare le variabili di fase. Per ulteriori informazioni, consulta the section called “Utilizzo delle variabili di fase”.
7. Scegli la scheda Canary, quindi seleziona Crea Canary. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda Canary.
8. In Impostazioni di Canary, inserisci in Canary la percentuale di richieste da deviare al Canary.
9. Se lo desideri, seleziona Cache della fase per attivare il caching per la release Canary. La cache non è disponibile per la versione Canary fino a quando la API memorizzazione nella cache non è abilitata.
10. Per sovrascrivere le variabili di fase esistenti, inserisci in Sostituzione canary un nuovo valore per la variabile di fase.

Dopo aver inizializzato la versione Canary nella fase di distribuzione, modificate API e desiderate testare le modifiche. È possibile API ridistribuirlo nella stessa fase in modo che sia la versione aggiornata che la versione base siano accessibili attraverso la stessa fase. Di seguito viene descritto come fare.

Per distribuire la API versione più recente su un canarino

1. Con ogni aggiornamento diAPI, scegli Deploy. API
2. In Deploy API, scegli lo stage che contiene un canarino dall'elenco a discesa Deploy stage.
3. (Facoltativo) Immetti una descrizione in Descrizione distribuzione.
4. Scegli Deploy per inviare la API versione più recente alla versione canary.
5. Se lo desideri, riconfigura le impostazioni della fase, i log o le impostazioni Canary, come descritto in Per creare la distribuzione di una release Canary iniziale.

Di conseguenza, la versione canaria punta alla versione più recente mentre la versione di produzione punta ancora alla versione iniziale di. API canarySettingsOra ha un nuovo deploymentIdvalore, mentre lo stage ha ancora il deploymentIdvalore iniziale. In background la console chiama stage:update.

Crea un dispiegamento canarino usando il AWS CLI

Crea prima di tutto una distribuzione di base con due variabili di fase, ma senza alcuna release Canary:

```
aws apigateway create-deployment \
    --variables sv0=val0,sv1=val1 \
    --rest-api-id abcd1234 \
    --stage-name 'prod' \
```

Il comando restituisce una rappresentazione dell'oggetto `Deployment` risultante, simile a quanto segue:

```
{
    "id": "du4ot1",
    "createdDate": 1511379050
}
```

La distribuzione risultante `id` identifica un'istantanea (o versione) di. API

Crea ora una distribuzione Canary nella fase `prod`:

```
aws apigateway create-deployment  --rest-api-id abcd1234 \
    --canary-settings \
    '{
        "percentTraffic":10.5,
        "useStageCache":false,
        "stageVariableOverrides":{
            "sv1":"val2",
            "sv2":"val3"
        }
    }' \
    --stage-name 'prod'
```

Se la fase specificata (`prod`) non esiste, il comando precedente restituisce un errore. In caso contrario, restituisce la rappresentazione della risorsa di distribuzione appena creata, simile a quanto segue:

```
{
    "id": "a6rox0",
    "createdDate": 1511379433
}
```

La distribuzione risultante `id` identifica la versione di prova di API per la versione canaria. Di conseguenza, la fase associata è abilitata per la release Canary. È possibile visualizzare la rappresentazione della fase chiamando il comando `get-stage`, in modo analogo a quanto segue:

```
aws apigateway get-stage --rest-api-id acbd1234 --stage-name prod
```

Di seguito è illustrata una rappresentazione dell'oggetto `Stage` come output del comando:

```
{
    "stageName": "prod",
    "variables": {
        "sv0": "val0",
        "sv1": "val1"
    },
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": "du4ot1",
    "lastUpdatedDate": 1511379433,
```

```
    "createdDate": 1511379050,
    "canarySettings": {
        "percentTraffic": 10.5,
        "deploymentId": "a6rox0",
        "useStageCache": false,
        "stageVariableOverrides": {
            "sv2": "val3",
            "sv1": "val2"
        }
    },
    "methodSettings": {}
}
```

In questo esempio, la versione base di API utilizzerà le variabili di fase di`{"sv0":val0", "sv1":val1"}`, mentre la versione di test utilizza le variabili di fase di. `{"sv1":val2", "sv2":val3"}` Sia la release di produzione che la release Canary usano la stessa variabile di fase `sv1`, ma con valori diversi, `val1` e `val2`, rispettivamente. La variabile di fase `sv0` viene usata unicamente nella release di produzione e la variabile di fase `sv2` viene usata unicamente nella release Canary.

Puoi creare una distribuzione di una release Canary da una distribuzione standard esistente aggiornando la fase per abilitare la release Canary. A tale scopo, crea innanzitutto una distribuzione normale:

```
aws apigateway create-deployment \
    --variables sv0=val0,sv1=val1 \
    --rest-api-id abcd1234 \
    --stage-name 'beta'
```

Il comando restituisce una rappresentazione della distribuzione della versione di base:

```
{
    "id": "cifeiw",
    "createdDate": 1511380879
}
```

La fase beta associata non ha impostazioni Canary:

```
{
    "stageName": "beta",
    "variables": {
```

```
        "sv0": "val0",
        "sv1": "val1"
    },
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": "cifeiw",
    "lastUpdatedDate": 1511380879,
    "createdDate": 1511380879,
    "methodSettings": {}
}
```

Crea ora una nuova distribuzione di una release Canary collegando una release Canary alla fase:

```
aws apigateway update-stage \
    --rest-api-id abcd1234 \
    --stage-name 'beta' \
    --patch-operations '[{
            "op": "replace",
            "value": "0.0",
            "path": "/canarySettings/percentTraffic"
        }, {
            "op": "copy",
            "from": "/canarySettings/stageVariableOverrides",
            "path": "/variables"
        }, {
            "op": "copy",
            "from": "/canarySettings/deploymentId",
            "path": "/deploymentId"
        }]'
```

Una rappresentazione della fase aggiornata è simile a quanto segue:

```
{
    "stageName": "beta",
    "variables": {
        "sv0": "val0",
        "sv1": "val1"
    },
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": "cifeiw",
    "lastUpdatedDate": 1511381930,
    "createdDate": 1511380879,
```

```
    "canarySettings": {
        "percentTraffic": 10.5,
        "deploymentId": "cifeiw",
        "useStageCache": false,
        "stageVariableOverrides": {
            "sv2": "val3",
            "sv1": "val2"
        }
    },
    "methodSettings": {}
}
```

Poiché abbiamo appena abilitato un canary su una versione esistente diAPI, sia la release di produzione (`Stage`) che la versione canary (`canarySettings`) puntano alla stessa implementazione, ovvero alla stessa versione (`deploymentId`) di. API Dopo aver modificato la versione API e averla nuovamente implementata in questa fase, la nuova versione sarà inclusa nella versione canary, mentre la versione base rimarrà nella versione di produzione. Ciò si manifesta nell'evoluzione della fase quando il valore `deploymentId` nella release Canary viene aggiornato al valore `id` della nuova distribuzione e il valore `deploymentId` nella release di produzione rimane invariato.

Aggiornamento di una release Canary

Dopo la distribuzione di una release Canary, è possibile modificare la percentuale di traffico nella release Canary oppure abilitare o disabilitare l'uso di una cache di fase per ottimizzare le prestazioni di test. È anche possibile modificare le variabili di fase usate nella release Canary quando il contesto di esecuzione viene aggiornato. Per effettuare tali aggiornamenti, chiamate l'operazione stage:update con i nuovi valori attivati. canarySettings

È possibile aggiornare una versione di Canary utilizzando la console API Gateway, il comando AWS CLI update-stage o un. AWS SDK

Argomenti

- Aggiorna una versione canaria utilizzando la console Gateway API
- Aggiorna una versione Canary utilizzando il AWS CLI

Aggiorna una versione canaria utilizzando la console Gateway API

Per utilizzare la console API Gateway per aggiornare le impostazioni Canary esistenti su uno stage, procedi come segue:

Per aggiornare le impostazioni di Canary esistenti

1. Accedi alla console API Gateway e scegline una esistente RESTAPI.
2. Nel riquadro di navigazione principale scegli Fasi, quindi seleziona una fase esistente.
3. Seleziona la scheda Canary, quindi scegli Modifica. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda Canary.
4. Aggiorna il campo Distribuzione richiesta aumentando o diminuendo il valore percentuale scegliendo un numero compreso tra 0,0 e 100,0 inclusi.
5. Seleziona o deseleziona la casella di controllo Cache della fase.
6. Aggiungi, rimuovi o modifica le Variabili di fase di Canary.
7. Seleziona Salva.

Aggiorna una versione Canary utilizzando il AWS CLI

Per utilizzare il AWS CLI per aggiornare un canarino, chiamate il `update-stage`comando.

Per abilitare o disabilitare l'uso di una cache di fase per la release Canary, esegui il comando `update-stage` come illustrato di seguito:

```
aws apigateway update-stage \
    --rest-api-id {rest-api-id} \
    --stage-name '{stage-name}'  \
    --patch-operations op=replace,path=/canarySettings/useStageCache,value=true
```

Per modificare la percentuale di traffico nella release Canary, esegui `update-stage` per sostituire il valore `/canarySettings/percentTraffic` nella fase.

```
aws apigateway update-stage \
    --rest-api-id {rest-api-id} \
    --stage-name '{stage-name}' \
    --patch-operations op=replace,path=/canarySettings/percentTraffic,value=25.0
```

Per aggiornare le variabili di fase Canary aggiungendo, sostituendo o rimuovendo una variabile:

```
aws apigateway update-stage  \
    --rest-api-id {rest-api-id} \
    --stage-name '{stage-name}'  \
    --patch-operations '[{
        "op": "replace",
```

```
        "path": "/canarySettings/stageVariableOverrides/newVar",
        "value": "newVal"
      }, {
        "op": "replace",
        "path": "/canarySettings/stageVariableOverrides/var2",
        "value": "val4"
      }, {
        "op": "remove",
        "path": "/canarySettings/stageVariableOverrides/var1"
      }]'
```

È possibile aggiornare tutti gli elementi precedenti combinando le operazioni in un singolo valore `patch-operations`:

```
aws apigateway update-stage  \
    --rest-api-id {rest-api-id} \
    --stage-name '{stage-name}' \
    --patch-operations '[{
        "op": "replace",
        "path": "/canarySettings/percentTraffic",
        "value": "20.0"
    }, {
        "op": "replace",
        "path": "/canarySettings/useStageCache",
        "value": "true"
    }, {
        "op": "remove",
        "path": "/canarySettings/stageVariableOverrides/var1"
    }, {
        "op": "replace",
        "path": "/canarySettings/stageVariableOverrides/newVar",
        "value": "newVal"
    }, {
        "op": "replace",
        "path": "/canarySettings/stageVariableOverrides/val2",
        "value": "val4"
      }]'
```

Promozione di una release Canary

Quando promuovi una versione canarina, questa sostituisce le impostazioni correnti dello stage. La promozione di una release Canary non comporta la disabilitazione della release Canary nella fase.

Per disabilitare una release Canary, è necessario rimuovere le impostazioni Canary nella fase. Per promuovere un canarino, procedi come segue.

- Reimpostazione dell'ID distribuzione della fase con le impostazioni dell'ID distribuzione della release Canary. Ciò aggiorna l'APIistantanea dello stage con l'istantanea del canarino, rendendo la versione di prova anche la versione di produzione.
- Aggiornamento delle variabili di fase con le variabili di fase della release Canary, se presenti. Questo aggiorna il contesto di API esecuzione dello stage con quello del canarino. Senza questo aggiornamento, la nuova API versione potrebbe produrre risultati imprevisti se la versione di test utilizza variabili di fase diverse o valori diversi delle variabili di fase esistenti.
- Impostazione della percentuale del traffico della release Canary su 0,0%.

Argomenti

- Promuovi una versione canary utilizzando la console API Gateway
- Promozione di una release Canary con AWS CLI

Promuovi una versione canary utilizzando la console API Gateway

Per utilizzare la console API Gateway per promuovere una distribuzione della versione Canary, procedi come segue:

Per promuovere l'implementazione di una release Canary

1. Accedi alla console API Gateway e scegline una esistente API nel pannello di navigazione principale.
2. Nel riquadro di navigazione principale scegli Fasi, quindi seleziona una fase esistente.
3. Scegli la scheda Canary.
4. Scegli Promuovi Canary.
5. Verifica le modifiche da apportare e scegli Promuovi Canary.

Dopo la promozione, la versione di produzione fa riferimento alla stessa API versione (deploymentId) della versione Canary. È possibile verificarlo utilizzando il AWS CLI. Per un esempio, consulta the section called “Promozione di una release Canary con AWS CLI”.

Promozione di una release Canary con AWS CLI

Per promuovere una versione canary alla versione di produzione utilizzando AWS CLI i comandi, chiamate il `update-stage` comando per copiare il file canary-associato `deploymentId` a quello associato allo stage`deploymentId`, per reimpostare la percentuale di traffico canarino su zero (`0.0`) e per copiare qualsiasi variabile dello stage canary-bound su quelle corrispondenti.

Supponiamo di avere una distribuzione di rilascio di Canary, descritta da una fase simile alla seguente:

```
{
    "_links": {
        ...
    },
    "accessLogSettings": {
        ...
    },
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "canarySettings": {
        "deploymentId": "eh1sby",
        "useStageCache": false,
        "stageVariableOverrides": {
            "sv2": "val3",
            "sv1": "val2"
        },
        "percentTraffic": 10.5
    },
    "createdDate": "2017-11-20T04:42:19Z",
    "deploymentId": "nfcn0x",
    "lastUpdatedDate": "2017-11-22T00:54:28Z",
    "methodSettings": {
        ...
    },
    "stageName": "prod",
    "variables": {
        "sv1": "val1"
    }
}
```

Per la promozione, chiamiamo la richiesta `update-stage` seguente:

```
aws apigateway update-stage  \
```

```
    --rest-api-id {rest-api-id}  \
    --stage-name '{stage-name}'  \
    --patch-operations '[{
        "op": "replace",
        "value": "0.0",
        "path": "/canarySettings/percentTraffic"
      }, {
        "op": "copy",
        "from": "/canarySettings/stageVariableOverrides",
        "path": "/variables"
      }, {
        "op": "copy",
        "from": "/canarySettings/deploymentId",
        "path": "/deploymentId"
      }]'
```

Dopo la promozione, la fase ora è come la seguente:

```
{
    "_links": {
        ...
    },
    "accessLogSettings": {
        ...
    },
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "canarySettings": {
        "deploymentId": "eh1sby",
        "useStageCache": false,
        "stageVariableOverrides": {
            "sv2": "val3",
            "sv1": "val2"
        },
        "percentTraffic": 0
    },
    "createdDate": "2017-11-20T04:42:19Z",
    "deploymentId": "eh1sby",
    "lastUpdatedDate": "2017-11-22T05:29:47Z",
    "methodSettings": {
        ...
    },
    "stageName": "prod",
```

```
    "variables": {
        "sv2": "val3",
        "sv1": "val2"
    }
}
```

Promuovere una versione Canary Release sullo stage non disabilita la Canary e la distribuzione rimane una distribuzione canary release. Per renderla una normale distribuzione di produzione, è necessario disabilitare le impostazioni Canary. Per ulteriori informazioni su come disabilitare una distribuzione di release Canary, consulta the section called “Disabilitazione di una release Canary”.

Disabilitazione di una release Canary

Per disabilitare l'implementazione di una release Canary, è necessario impostare `canarySettings` su null per rimuoverla dalla fase.

È possibile disabilitare una distribuzione Canary Release utilizzando la console API Gateway, il AWS CLI, o un. AWS SDK

Argomenti

- Disattiva una versione canary utilizzando la API console Gateway
- Disattivazione di una release Canary con la AWS CLI

Disattiva una versione canary utilizzando la API console Gateway

Per utilizzare la console API Gateway per disattivare una distribuzione di Canary Release, procedi nel seguente modo:

Per disabilitare l'implementazione di una release Canary

1. Accedi alla console API Gateway e scegline una esistente API nel pannello di navigazione principale.
2. Nel riquadro di navigazione principale scegli Fasi, quindi seleziona una fase esistente.
3. Scegli la scheda Canary.
4. Scegli Elimina.
5. Confermare che si desidera eliminare la release Canary scegliendo Delete (Elimina).

Di conseguenza, la proprietà `canarySettings` diventa `null` e viene rimossa dalla fase di distribuzione. È possibile verificarlo utilizzando il AWS CLI. Per un esempio, consulta the section called “Disattivazione di una release Canary con la AWS CLI”.

Disattivazione di una release Canary con la AWS CLI

Per utilizzare il AWS CLI per disattivare una distribuzione in versione canaria, chiama il `update-stage` comando come segue:

```
aws apigateway update-stage \
    --rest-api-id abcd1234 \
    --stage-name canary \
    --patch-operations '[{"op":"remove", "path":"/canarySettings"}]'
```

Una risposta corretta restituisce un payload simile al seguente:

```
{
    "stageName": "prod",
    "accessLogSettings": {
        ...
    },
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": "nfcn0x",
    "lastUpdatedDate": 1511309280,
    "createdDate": 1511152939,
    "methodSettings": {
        ...
    }
}
```

Come illustrato nell'output, la proprietà `canarySettings` non è più presente nella fase di una distribuzione disabilitata della release Canary.

Gli aggiornamenti REST APIs che richiedono una ridistribuzione

Il mantenimento di API un significa visualizzare, aggiornare ed eliminare le configurazioni esistenti. API È possibile mantenerla API utilizzando la console API Gateway AWS CLI, un SDK o il API Gateway. REST API L'aggiornamento di un API comporta la modifica di determinate proprietà delle risorse o delle impostazioni di configurazione diAPI. Gli aggiornamenti delle risorse richiedono la ridistribuzione diAPI, mentre gli aggiornamenti di configurazione non lo richiedono.

La tabella seguente descrive API le risorse che richiedono la ridistribuzione delle risorse al API momento dell'aggiornamento.

Risorsa	Note
ApiKey	Per le proprietà applicabili e le operazioni supportate, consulta apikey:update. L'aggiornamento richiede la ridistribuzione di. API
Authorizer	Per le proprietà applicabili e le operazioni supportate, consulta authorizer:update. L'aggiornamento richiede la ridistribuzione di. API
DocumentationPart	Per le proprietà applicabili e le operazioni supportate, consulta documentationpart:update. L'aggiornamento richiede la ridistribuzione di. API
DocumentationVersion	Per le proprietà applicabili e le operazioni supportate, consulta documentationversion:update. L'aggiornamento richiede la ridistribuzione di. API
GatewayResponse	Per le proprietà applicabili e le operazioni supportate, consulta gatewayresponse:update. L'aggiornamento richiede la ridistribuzione di. API
Integration	Per le proprietà applicabili e le operazioni supportate, consulta integration:update. L'aggiornamento richiede la ridistribuzione di. API
IntegrationResponse	Per le proprietà applicabili e le operazioni supportate, consulta integrationresponse:update. L'aggiornamento richiede la ridistribuzione di. API
Metodo	Per le proprietà applicabili e le operazioni supportate, consulta method:update. L'aggiornamento richiede la ridistribuzione di. API
MethodResponse	Per le proprietà applicabili e le operazioni supportate, consulta methodresponse:update. L'aggiornamento richiede la ridistribuzione di. API
Modello	Per le proprietà applicabili e le operazioni supportate, consulta model:update. L'aggiornamento richiede la ridistribuzione di. API
RequestValidator	Per le proprietà applicabili e le operazioni supportate, consulta requestvalidator:update. L'aggiornamento richiede la ridistribuzione di. API
Resource (Risorsa)	Per le proprietà applicabili e le operazioni supportate, consulta resource:update. L'aggiornamento richiede la ridistribuzione di. API

Risorsa	Note
RestApi	Per le proprietà applicabili e le operazioni supportate, consulta restapi:update. L'aggiornamento richiede la ridistribuzione di. API
VpcLink	Per le proprietà applicabili e le operazioni supportate, consulta vpclink:update. L'aggiornamento richiede la ridistribuzione di. API

La tabella seguente descrive API le configurazioni che non richiedono la ridistribuzione delle impostazioni al API momento dell'aggiornamento.

Configurazione	Note
Account	Per le proprietà applicabili e le operazioni supportate, consulta account:update. L'aggiornamento non richiede la ridistribuzione di. API
Distribuzione	Per le proprietà applicabili e le operazioni supportate, consulta deployment:update.
DomainName	Per le proprietà applicabili e le operazioni supportate, consulta domainname:update. L'aggiornamento non richiede la ridistribuzione di. API
BasePathMapping	Per le proprietà applicabili e le operazioni supportate, consulta basepathmapping:update. L'aggiornamento non richiede la ridistribuzione di. API
Fase	Per le proprietà applicabili e le operazioni supportate, consulta stage:update. L'aggiornamento non richiede la ridistribuzione di. API
Utilizzo	Per le proprietà applicabili e le operazioni supportate, consulta usage:update. L'aggiornamento non richiede la ridistribuzione di. API
UsagePlan	Per le proprietà applicabili e le operazioni supportate, consulta usageplan:update. L'aggiornamento non richiede la ridistribuzione di. API

Nome di dominio personalizzato per REST APIs in API Gateway

I nomi di dominio personalizzati sono più semplici e intuitivi URLs che puoi fornire ai tuoi API utenti.

Dopo aver distribuito il tuoAPI, tu (e i tuoi clienti) potete richiamarlo API utilizzando la base predefinita URL del seguente formato:

```
https://api-id.execute-api.region.amazonaws.com/stage
```

dove *`api-id`* è generato da API Gateway, *`region`* è la AWS regione, e *`stage`* è specificata dall'utente durante la API distribuzione di.

La parte hostname diURL, si *`api-id`*`.execute-api.`*`region`*`.amazonaws.com` riferisce a un API endpoint. Il nome dell'APIendpoint predefinito viene generato casualmente, è difficile da richiamare e non è intuitivo.

Con i nomi di dominio personalizzati, puoi configurare il tuo hostname e scegliere un percorso API di base (ad esempio,`myservice`) per mappare l'alternativa al tuo. URL API Ad esempio, una API base più intuitiva URL può diventare:

```
https://api.example.com/myservice
```

Considerazioni

Le seguenti considerazioni potrebbero influire sull'utilizzo di un nome di dominio personalizzato.

- I nomi di dominio personalizzati non sono supportati per uso privatoAPIs.
- Puoi disabilitare l'endpoint predefinito per il tuoAPI. I client possono comunque connettersi all'endpoint predefinito, ma riceveranno un codice di stato `403 Forbidden`.
- Un nome di dominio personalizzato regionale può essere associato a REST APIs e HTTPAPIs. È possibile utilizzare la versione 2 del API Gateway APIs per creare e gestire nomi di dominio personalizzati regionali per RESTAPIs.
- Un nome di dominio personalizzato deve essere univoco all'interno di una regione per tutti gli AWS account.
- Puoi eseguire la migrazione di un nome di dominio personalizzato tra endpoint ottimizzati per edge ed endpoint regionali.
- È necessario creare o aggiornare il record di risorse del DNS provider da mappare all'APIendpoint. Senza tale mappatura, API le richieste associate al nome di dominio personalizzato non possono raggiungere API Gateway.

- È possibile supportare un numero quasi infinito di nomi di dominio senza superare la quota predefinita utilizzando un certificato wildcard. Per ulteriori informazioni, consulta the section called “Nomi di dominio personalizzati con caratteri jolly”.
- Puoi scegliere una politica di sicurezza per il tuo nome di dominio personalizzato. Per ulteriori informazioni, consulta the section called “Scegli una politica di sicurezza”.
- Per configurare API mappature con più livelli, è necessario utilizzare un nome di dominio personalizzato regionale e utilizzare la politica di sicurezza TLS 1.2.

Prerequisiti

Di seguito sono riportati i prerequisiti per la creazione di un nome di dominio personalizzato.

Registrare un nome di dominio

È necessario disporre di un nome di dominio Internet registrato per configurare nomi di dominio personalizzati per il proprioAPIs. Puoi registrare il tuo nome di dominio Internet utilizzando Amazon Route 53 o utilizzando un registrar di domini di terze parti a tua scelta. Il tuo nome di dominio personalizzato può essere il nome di un sottodominio o il dominio principale (noto anche come «apice di zona») di un dominio Internet registrato.

Il nome di dominio deve seguire la specifica RFC1035 e può avere un massimo di 63 ottetti per etichetta e 255 ottetti in totale.

Specificate il certificato per il vostro nome di dominio personalizzato

Prima di configurare un nome di dominio personalizzato per unAPI, è necessario disporre di un TLS certificatoSSL/ACM. Se non ACM è disponibile nella AWS regione in cui si sta creando il nome di dominio personalizzato, è necessario importare un certificato in API Gateway in quella regione.

Per importare un TLS certificatoSSL/, è necessario fornire il corpo del TLS certificato PEM SSL / formattato, la relativa chiave privata e la catena di certificati per il nome di dominio personalizzato.

Ogni certificato memorizzato in ACM è identificato dal relativo. ARN Con i certificati emessi daACM, non devi preoccuparti di esporre dettagli sensibili del certificato, come la chiave privata. Per utilizzare un certificato AWS gestito per un nome di dominio, è sufficiente fare riferimento al relativoARN.

Se l'applicazione utilizza il blocco dei certificati, a volte noto come SSL pinning, per bloccare un ACM certificato, l'applicazione potrebbe non essere in grado di connettersi al dominio dopo il AWS rinnovo del certificato. Per ulteriori informazioni, consulta Probemi nel blocco dei certificati nella Guida per l'utente di AWS Certificate Manager .

Nomi di dominio personalizzati con caratteri jolly

Con i nomi di dominio personalizzati con caratteri jolly, è possibile supportare un numero quasi infinito di nomi di dominio senza superare la quota predefinita. Ad esempio, è possibile dare a ciascuno dei clienti il proprio nome di dominio, `customername.api.example.com`.

Per creare un nome di dominio personalizzato con caratteri jolly, specificare un carattere jolly (`*`) come primo sottodominio di un dominio personalizzato che rappresenta tutti i possibili sottodomini di un dominio radice.

Ad esempio, il nome di dominio personalizzato con caratteri jolly `*.example.com` genera sottodomini quali `a.example.com`, `b.example.com` e `c.example.com`, indirizzati tutti allo stesso dominio.

I nomi di dominio personalizzati Wildcard supportano configurazioni diverse dai nomi di dominio personalizzati standard di API Gateway. Ad esempio, in un singolo AWS account, è possibile configurare `*.example.com` e `a.example.com` comportarsi in modo diverso.

Puoi utilizzare le variabili di `$context.domainPrefix` contesto `$context.domainName` e per determinare il nome di dominio utilizzato da un client per chiamare il tuoAPI. Per ulteriori informazioni sulle variabili di contesto, consulta APIModello di mappatura del gateway e riferimento alla variabile di registrazione degli accessi.

Per creare un nome di dominio personalizzato con caratteri jolly, devi fornire un certificato emesso da ACM che sia stato convalidato utilizzando il metodo di convalida DNS o tramite e-mail.

> **Note**
>
> Non puoi creare un nome di dominio personalizzato con caratteri jolly se un altro AWS account ha creato un nome di dominio personalizzato che è in conflitto con il nome di dominio personalizzato con caratteri jolly. Ad esempio, se l'account A ha creato `a.example.com`, l'account B non può creare il nome di dominio personalizzato con caratteri jolly `*.example.com`.
> Se l'account A e l'account B condividono un proprietario, è possibile contattare il Centro assistenza AWS per richiedere un'eccezione.

Passaggi successivi per i nomi di dominio personalizzati

Di seguito sono riportati i passaggi successivi per i nomi di dominio personalizzati:

Passaggi successivi

- Per informazioni su come impostare il tuo TLS certificatoSSL/, consultathe section called “Prepara i certificati in AWS Certificate Manager”.
- Per informazioni su come creare un nome di dominio personalizzato regionale, consultathe section called “Configura un nome di dominio personalizzato regionale”.
- Per informazioni su come creare un nome di dominio personalizzato ottimizzato per i dispositivi edge, consulta. the section called “Configura un nome di dominio personalizzato ottimizzato per i dispositivi periferici in Gateway API”
- Per informazioni su come migrare tra nomi di dominio personalizzati regionali e ottimizzati per i dispositivi periferici, consulta. the section called “Migrazione di nomi di dominio personalizzati”
- Per informazioni su come collegare API Stages a un nome di dominio personalizzato, consulta. the section called “APImappature”
- Per informazioni su come scegliere una politica di sicurezza per il nome di dominio personalizzato, consultathe section called “Scegli una politica di sicurezza”.
- Per informazioni su come disattivare l'endpoint predefinito per il nome di dominio personalizzato, consultathe section called “Disabilitazione dell'endpoint predefinito”.
- Per informazioni su come utilizzare i controlli di integrità di Route 53 per controllare il DNS failover da un API gatewayAPI, consulta. the section called “DNSfailover”

Se è la prima volta che crei un nome di dominio personalizzato, ti consigliamo di iniziare conthe section called “Prepara i certificati in AWS Certificate Manager”, specificare il certificato e quindi the section called “Configura un nome di dominio personalizzato regionale” creare un nome di dominio personalizzato regionale.

Prepara i certificati in AWS Certificate Manager

Prima di configurare un nome di dominio personalizzato per unAPI, devi avere un TLS certificatoSSL/ pronto in AWS Certificate Manager. Per ulteriori informazioni, consulta la Guida per l'utente AWS Certificate Manager.

Considerazioni

Di seguito sono riportate alcune considerazioni relative al TLS certificato SSL /.

- Se crei un nome di dominio personalizzato ottimizzato per i dispositivi edge, API Gateway lo sfrutta CloudFront per supportare i certificati per i nomi di dominio personalizzati. Pertanto, i requisiti e

i vincoli di un nome SSL di dominio/certificato personalizzato sono dettati da. TLS CloudFront La dimensione massima della chiave pubblica è ad esempio di 2048, mentre la dimensione della chiave privata può essere di 1024, 2048 e 4096. La dimensione della chiave pubblica è determinata dall'autorità di certificazione usata. Chiedi all'autorità di certificazione di restituire chiavi di dimensioni diverse dalla lunghezza predefinita. Per ulteriori informazioni, consulta Accesso sicuro agli oggetti e Creazione di cookie firmati URLs e firmati.

- Per utilizzare un ACM certificato con un nome di dominio personalizzato regionale, devi richiedere o importare il certificato nella stessa regione della tuaAPI. Il certificato deve essere firmato da un'autorità di certificazione attendibile pubblicamente e coprire il nome dominio personalizzato.
- Per utilizzare un ACM certificato con un nome di dominio personalizzato ottimizzato per i dispositivi edge, devi richiedere o importare il certificato nella regione Stati Uniti orientali (Virginia settentrionale). `us-east-1`
- È necessario disporre di un nome di dominio registrato, ad esempio. *`example.com`* È possibile usare Amazon Route 53 o un registrar di dominio di terze parti accreditato. Per un elenco di tali registrar, consulta l'Accredited Registrar Directory sul sito web. ICANN

Per creare o importare un certificato/in SSL TLS ACM

Le seguenti procedure mostrano come creare o importare un TLS certificatoSSL/per un nome di dominio.

To request a certificate provided by ACM for a domain name

1. Accedere alla console AWS Certificate Manager.
2. Scegliere Request a certificate (Richiedi un certificato).
3. Per Tipo di certificato, scegli Richiedi un certificato pubblico.
4. Scegli Next (Successivo).
5. Per Nome di dominio completo, inserisci un nome di dominio personalizzato perAPI, ad esempio,`api.example.com`.
6. Facoltativamente, scegliere Add another name to this certificate (Aggiungi un altro nome a questo certificato).
7. Per Metodo di convalida, scegli un metodo per convalidare la proprietà del dominio.
8. Per Algoritmo chiave, scegli un algoritmo di crittografia.
9. Scegli Richiedi.

10. Affinché una richiesta sia valida, il proprietario registrato del dominio Internet deve acconsentire alla richiesta prima di ACM emettere il certificato. Se utilizzi Route 53 per gestire i tuoi DNS record pubblici, puoi aggiornarli direttamente tramite la ACM console.

To import into ACM a certificate for a domain name

1. Ottieni un TLS certificato PEM -encodedSSL/per il tuo nome di dominio personalizzato da un'autorità di certificazione. Per un elenco parziale di questiCAs, consulta la Mozilla Included CA List.

   a. Genera una chiave privata per il certificato e salva l'output in un file, utilizzando il SSL toolkit Open sul sito web di OpenSSL:

   ```
   openssl genrsa -out private-key-file 2048
   ```

   b. Genera una richiesta di firma del certificato (CSR) con la chiave privata generata in precedenza, utilizzando OpenSSL:

   ```
   openssl req -new -sha256 -key private-key-file -out CSR-file
   ```

   c. Invia il CSR file all'autorità di certificazione e salva il certificato risultante.

   d. Scarica la catena di certificati dall'autorità di certificazione.

   > **Note**
   >
   > Se si ottiene la chiave privata in un altro modo e la chiave è crittografata, è possibile utilizzare il seguente comando per decrittografare la chiave prima di inviarla a API Gateway per la configurazione di un nome di dominio personalizzato.
   >
   > ```
   > openssl pkcs8 -topk8 -inform pem -in MyEncryptedKey.pem -outform pem -
   > nocrypt -out MyDecryptedKey.pem
   > ```

2. Carica il certificato su: AWS Certificate Manager

   a. Accedere alla console AWS Certificate Manager.

   b. Seleziona Import a certificate (Importa un certificato).

c. Per Organismo del certificato, inserisci il corpo del certificato del server in PEM formato elettronico rilasciato dall'autorità di certificazione. Di seguito è illustrato un esempio abbreviato di tale certificato.

```
-----BEGIN CERTIFICATE-----
EXAMPLECA+KgAwIBAgIQJ1XxJ8Pl++gOfQtj0IBoqDANBgkqhkiG9w0BAQUFADBB
...
az8Cg1aicxLBQ7EaWIhhgEXAMPLE
-----END CERTIFICATE-----
```

d. Per la chiave privata del certificato, inserisci la chiave privata del certificato PEM -formatted. Di seguito è illustrato un esempio abbreviato di tale chiave.

```
-----BEGIN RSA PRIVATE KEY-----
EXAMPLEBAAKCAQEA2Qb3LDHD7StY7Wj6U2/opV6Xu37qUCCkeDWhwpZMYJ9/nETO
...
1qGvJ3u04vdnzaYN5WoyN5LFckrlA71+CszD1CGSqbVDWEXAMPLE
-----END RSA PRIVATE KEY-----
```

e. Per Certificate chain, inserisci i certificati intermedi PEM -formatted e, facoltativamente, il certificato principale, uno dopo l'altro senza righe vuote. Se includi il certificato root, la catena di certificati deve iniziare con i certificati intermedi e terminare con il certificato root. Usa i certificati intermedi forniti dall'autorità di certificazione. Non includere intermediari non presenti nella catena del percorso di trust. Di seguito è illustrato un esempio abbreviato.

```
-----BEGIN CERTIFICATE-----
EXAMPLECA4ugAwIBAgIQWrYdrB5NogYUx1U9Pamy3DANBgkqhkiG9w0BAQUFADCB
...
8/ifBlIK3se2e4/hEfcEejX/arxbx1BJCHBvlEPNnsdw8EXAMPLE
-----END CERTIFICATE-----
```

Ecco un altro esempio.

```
-----BEGIN CERTIFICATE-----
Intermediate certificate 2
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Intermediate certificate 1
-----END CERTIFICATE-----
```

```
-----BEGIN CERTIFICATE-----
Optional: Root certificate
-----END CERTIFICATE-----
```

f. Scegli Successivo e di nuovo Successivo.

Dopo che il certificato è stato creato o importato correttamente, prendi nota del certificato. ARN Sarà necessario in seguito per la configurazione del nome di dominio personalizzato.

Configura un nome di dominio personalizzato regionale in API Gateway

Usa un nome di dominio personalizzato regionale per creare una API base URL intuitiva. Con un nome di dominio personalizzato regionale, puoi mappare HTTP e utilizzare REST API lo stesso nome di dominio personalizzato e utilizzare l'TLSautenticazione reciproca.

Considerazioni

Di seguito sono riportate alcune considerazioni relative al nome di dominio personalizzato regionale.

- È necessario fornire un certificato specifico per la regione. ACM Questo certificato deve essere nella stessa regione della tua. API Per ulteriori informazioni sulla creazione o il caricamento di un certificato di un nome di dominio personalizzato consulta the section called “Prepara i certificati in AWS Certificate Manager”.
- Quando crei un nome di dominio personalizzato regionale (o ne migri uno) con un ACM certificato, API Gateway crea un ruolo collegato al servizio nel tuo account. Il ruolo collegato al servizio è necessario per allegare il ACM certificato all'endpoint regionale. Il ruolo è denominato AWSServiceRoleForAPIGatewaye avrà la politica APIGatewayServiceRolePolicygestita allegata. Per ulteriori informazioni sull'uso del ruolo collegato ai servizi, consulta Utilizzo dei ruoli collegati ai servizi.
- Dopo aver creato il nome di dominio personalizzato regionale, è necessario creare un DNS record per indirizzare il nome di dominio personalizzato al nome di dominio regionale. Ciò consente al traffico associato al nome di dominio personalizzato di essere indirizzato al nome API host regionale. Il DNS record può essere di tipo CNAME o «A».

Crea un nome di dominio personalizzato regionale

La procedura seguente mostra come creare un nome di dominio personalizzato regionale. Dopo aver completato questa procedura, è necessario creare un DNS record per indirizzare il nome di dominio personalizzato al nome di dominio regionale.

AWS Management Console

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere Custom Domain Names (Nomi di dominio personalizzati) nel riquadro di navigazione principale.
3. Seleziona Crea.
4. In Domain name (Nome di dominio), immettere un nome di dominio.
5. Per TLSVersione minima, seleziona una versione.
6. In Configurazione dell'endpoint, per il tipo di API endpoint, scegli Regionale.
7. Scegliere un certificato ACM. Il certificato deve trovarsi nella stessa regione del. API
8. Scegli Create (Crea) .

AWS CLI

Il create-domain-namecomando seguente può essere utilizzato per creare un nome di dominio personalizzato:

```
aws apigatewayv2 create-domain-name \
    --domain-name 'regional.example.com' \
    --domain-name-configurations CertificateArn=arn:aws:acm:us-
west-2:123456789012:certificate/123456789012-1234-1234-1234-12345678
```

Se l'operazione va a buon fine, la risposta alla chiamata presenta un risultato simile al seguente:

```
{
    "ApiMappingSelectionExpression": "$request.basepath",
    "DomainName": "regional.example.com",
    "DomainNameConfigurations": [
        {
            "ApiGatewayDomainName": "d-numh1z56v6.execute-api.us-
west-2.amazonaws.com",
            "CertificateArn": "arn:aws:acm:us-
west-2:123456789012:certificate/123456789012-1234-1234-1234-12345678",
            "DomainNameStatus": "AVAILABLE",
            "EndpointType": "REGIONAL",
            "HostedZoneId": "Z2OJLYMUO9EFXC",
            "SecurityPolicy": "TLS_1_2"
        }
```

```
    ]
}
```

Il valore della `DomainNameConfigurations` proprietà restituisce il nome host API del Regional. È necessario creare un DNS record per indirizzare il nome di dominio personalizzato a questo nome di dominio regionale. Ciò consente al traffico associato al nome di dominio personalizzato di essere indirizzato al nome host API di questa regione.

Crea un DNS record per il tuo nome di dominio personalizzato regionale

La procedura seguente mostra come creare un DNS record che indichi il nome di dominio personalizzato al nome di dominio regionale appena creato.

AWS Management Console

Per utilizzare il AWS Management Console, segui la documentazione di Route 53 sulla configurazione di Route 53 per instradare il traffico verso API Gateway.

AWS CLI

Per configurare i DNS record in modo da mappare il nome di dominio personalizzato regionale al nome host dell'ID della zona ospitata specificato, crea innanzitutto un JSON file che contenga la configurazione per l'impostazione di un DNS record per il nome di dominio regionale.

Di seguito `setup-dns-record.json` viene illustrato come creare un DNS A record per mappare un nome di dominio personalizzato regionale (`regional.example.com`) al relativo hostname regionale (`d-numh1z56v6.execute-api.us-west-2.amazonaws.com`) fornito come parte della creazione del nome di dominio personalizzato. Le proprietà `DNSName` e `HostedZoneId` di `AliasTarget` possono prendere rispettivamente i valori `regionalDomainName` e `regionalHostedZoneId` del nome di dominio personalizzato. Puoi anche ottenere la Regional Route 53 Hosted Zone IDs in Amazon API Gateway Endpoints and Quotas.

```
{
  "Changes": [
    {
      "Action": "CREATE",
      "ResourceRecordSet": {
        "Name": "regional.example.com",
        "Type": "A",
```

```
        "AliasTarget": {
          "DNSName": "d-numh1z56v6.execute-api.us-west-2.amazonaws.com",
          "HostedZoneId": "Z2OJLYMUO9EFXC",
          "EvaluateTargetHealth": false
        }
      }
    }
  ]
}
```

Di seguito change-resource-record-setsviene illustrato come creare un DNS record per il nome di dominio personalizzato regionale:

```
aws route53 change-resource-record-sets \
    --hosted-zone-id Z2OJLYMUO9EFXC \
    --change-batch file://path/to/your/setup-dns-record.json
```

`hosted-zone-id`Sostituiscilo con l'ID della zona ospitata Route 53 del DNS record impostato nel tuo account. Il valore del `change-batch` parametro punta a un JSON file (*`setup-dns-record.json`*) in una cartella (*`path/to/your`*).

Crea una mappatura del percorso di base per il tuo nome di dominio personalizzato regionale

Dopo aver creato un DNS record, crei una mappatura del percorso di base per mappare il tuo API al nome di dominio personalizzato. Ad esempio, puoi mappare lo stage `test` per l'APIID di `abc123` al nome di dominio personalizzato `regional.example.com` utilizzando la chiave di API mappatura di. `myApi` Questo mappa il APIhttps`://abcd1234.execute-api.us-west-2.amazonaws.com/test`. `https://regional.example.com/myApi`

AWS Management Console

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere un nome di dominio personalizzato.
3. Scegli Configura mappature. API
4. Scegliere Add new mapping (Aggiungi nuova mappatura).
5. Specificate APIlo stadio e il percorso per la mappatura.
6. Seleziona Salva.

AWS CLI

Il create-api-mappingcomando seguente può essere usato per creare una mappatura del percorso di base:

```
aws apigatewayv2 create-api-mapping \
    --domain-name 'regional.example.com' \
    --api-mapping-key 'myApi' \
    --api-id abcd1234 \
    --stage 'test'
```

Di conseguenza, la base che URL utilizza il nome di dominio personalizzato per il API che viene distribuito nella fase diventa. `https://regional.example.com/myAPI`

Con un nome di dominio personalizzato regionale, puoi creare una API mappatura con più livelli, ad esempio. `https://regional.example.com/orders/v1/items/123` Puoi anche mappare HTTP e REST API utilizzare lo stesso nome di dominio personalizzato. Per ulteriori informazioni, consulta the section called “APImappature”.

Configura un nome di dominio personalizzato ottimizzato per i dispositivi periferici in Gateway API

Quando si crea un nome di dominio personalizzato per un sistema ottimizzato per i dispositivi edgeAPI, API Gateway imposta una CloudFront distribuzione e un DNS record per mappare il nome di API dominio al nome di dominio di CloudFront distribuzione. Le richieste per il API vengono quindi indirizzate a API Gateway tramite la distribuzione mappata. CloudFront Questa mappatura riguarda API le richieste destinate al nome di dominio personalizzato da indirizzare a API Gateway tramite la distribuzione mappata. CloudFront

Considerazioni

Di seguito sono riportate alcune considerazioni relative al nome di dominio personalizzato ottimizzato per i dispositivi edge.

- Per configurare un nome di dominio personalizzato ottimizzato per i dispositivi periferici o per aggiornarne il certificato, devi disporre dell'autorizzazione per aggiornare le distribuzioni. CloudFront

  Per aggiornare le distribuzioni sono necessarie le seguenti autorizzazioni: CloudFront

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Sid": "AllowCloudFrontUpdateDistribution",
            "Effect": "Allow",
            "Action": [
                "cloudfront:updateDistribution"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

- È necessario richiedere o importare un certificato per il nome di dominio personalizzato ottimizzato per i dispositivi edge nella regione Stati Uniti orientali (Virginia settentrionale). `us-east-1`
- La CloudFront distribuzione creata da API Gateway è di proprietà di un account specifico della regione affiliato a Gateway. API Quando si tracciano le operazioni per creare e aggiornare tale CloudFront distribuzione CloudTrail, è necessario utilizzare questo API ID account Gateway. Per ulteriori informazioni, consulta Accedi alla creazione di nomi di dominio personalizzati CloudTrail.
- APIGateway supporta nomi di dominio personalizzati ottimizzati per i dispositivi perimetrali sfruttando Server Name Indication () SNI sulla distribuzione. CloudFront Per ulteriori informazioni sull'utilizzo di nomi di dominio personalizzati su una CloudFront distribuzione, incluso il formato di certificato richiesto e la dimensione massima della lunghezza della chiave del certificato, consulta Using Alternate Domain Names e HTTPS nella Amazon CloudFront Developer Guide
- Un nome di dominio personalizzato ottimizzato per l'edge impiega circa 40 minuti per essere pronto.
- Dopo aver creato il nome di dominio personalizzato ottimizzato per i dispositivi perimetrali, è necessario creare un DNS record per mappare il nome di dominio personalizzato al nome di distribuzione. CloudFront

Crea un nome di dominio personalizzato ottimizzato per i bordi

La procedura seguente descrive come creare un nome di dominio personalizzato ottimizzato per i bordi per un. API

AWS Management Console

1. Accedere alla console API Gateway all'indirizzo /apigateway. https://console.aws.amazon.com
2. Scegliere Custom Domain Names (Nomi di dominio personalizzati) nel riquadro di navigazione principale.
3. Seleziona Crea.
4. In Domain name (Nome di dominio), immettere un nome di dominio.
5. Per TLSVersione minima, seleziona una versione.
6. In Configurazione dell'endpoint, per il tipo di API endpoint, scegli Ottimizzato per Edge.
7. Per il ACMcertificato, scegli un certificato. ACM
8. Scegliere Create domain name (Crea nome di dominio).

REST API

1. Chiama domainname:create, specificando il nome di dominio personalizzato e il certificato archiviato inARN. AWS Certificate Manager

   La API chiamata riuscita restituisce una `201 Created` risposta contenente il certificato e il nome ARN di CloudFront distribuzione associato nel relativo payload.
2. Annota il nome del dominio di CloudFront distribuzione mostrato nell'output. Ne hai bisogno nel passaggio successivo per impostare il CNAME valore del dominio personalizzato o la destinazione dell'alias A-record nel tuo. DNS

Per esempi di codice di questa REST API chiamata, vedi domainname:create.

Un nome di dominio personalizzato ottimizzato per l'edge impiega circa 40 minuti per essere pronto, ma la console visualizza immediatamente il nome di dominio di CloudFront distribuzione associato, sotto forma di, insieme al certificato. `distribution-id.cloudfront.net` ARN Nel frattempo, puoi continuare con il passaggio successivo e configurare l'alias del DNS record per mappare il nome di dominio personalizzato al nome di dominio di distribuzione associato. CloudFront

Crea un DNS record per il tuo nome di dominio personalizzato ottimizzato per edge

Dopo aver avviato la creazione del nome di dominio personalizzato ottimizzato per i dispositivi perimetrali, configura l'alias del record. DNS

Ti consigliamo di utilizzare Route 53 per creare un alias di record A per il tuo nome di dominio personalizzato e specificare il nome di dominio di CloudFront distribuzione come destinazione dell'alias. Ciò significa che Route 53 è in grado di instradare il nome di dominio personalizzato anche se si tratta di un apex di zona. Per ulteriori informazioni, consulta la pagina relativa alla scelta tra set di record della risorsa alias e non alias nella Guida per gli sviluppatori di Amazon Route 53.

Puoi invece aggiungere il tuo dominio personalizzato alla zona ospitata come set di record di CNAME risorse. Il nome del CNAME record specifica il nome di dominio personalizzato immesso in precedenza in Domain Name (ad esempio,`api.example.com`). Il valore del CNAME record specifica il nome di dominio per la CloudFront distribuzione.

Tuttavia, l'uso di un CNAME record non funzionerà se il dominio personalizzato è un apice di zona (ad esempio, `example.com` anzichéapi.example.com). Un apex di zona è anche noto comunemente come dominio root dell'organizzazione. Per un apice di zona è necessario utilizzare un alias di record A, a condizione che sia supportato dal provider. DNS

Per istruzioni su Amazon Route 53, consulta Routing del traffico verso un Amazon API Gateway API utilizzando il tuo nome di dominio nella Amazon Route 53 Developer Guide.

Accedi alla creazione di nomi di dominio personalizzati CloudTrail

Quando CloudTrail è abilitato per la registrazione delle chiamate API Gateway effettuate dall'account, API Gateway registra gli aggiornamenti di CloudFront distribuzione associati quando viene creato o aggiornato un nome di dominio personalizzato per un. API Questi registri sono disponibili in. `us-east-1` Poiché queste CloudFront distribuzioni sono di proprietà di API Gateway, ognuna di queste CloudFront distribuzioni segnalate è identificata da uno dei seguenti account API Gateway specifici della regioneIDs, anziché dall'ID dell'account del API proprietario.

Region	ID account
us-east-1	392220576650
us-east-2	718770453195
us-west-1	968246515281
us-west-2	109351309407
ca-central-1	796887884028

Region	ID account
eu-west-1	631144002099
eu-west-2	544388816663
eu-west-3	061510835048
eu-central-1	474240146802
eu-central-2	166639821150
eu-north-1	394634713161
eu-south-1	753362059629
eu-south-2	359345898052
ap-northeast-1	969236854626
ap-northeast-2	020402002396
ap-northeast-3	360671645888
ap-southeast-1	195145609632
ap-southeast-2	798376113853
ap-southeast-3	652364314486
ap-southeast-4	849137399833
ap-south-1	507069717855
ap-south-2	644042651268
ap-east-1	174803364771
sa-east-1	287228555773
me-south-1	855739686837

Region	ID account
me-central-1	614065512851

Configura la mappatura del percorso di base di un utente API con un nome di dominio personalizzato come nome host

È possibile utilizzare la mappatura del percorso di base per utilizzare un singolo nome di dominio personalizzato come nome host di più nomi. APIs Ciò lo rende API accessibile tramite la combinazione del nome di dominio personalizzato e del percorso di base associato.

Ad esempio, se in API Gateway hai creato un API nome `PetStore` e un altro API nome `Dogs` e poi hai impostato un nome di dominio personalizzato di`api.example.com`, puoi `PetStore` API impostarli URL come`https://api.example.com`.

Questo lo associa al `PetStore` API percorso di base di una stringa vuota. Se imposti i `PetStore` API s URL come`https://api.example.com/PetStore`, questo li associa al `PetStore` API percorso base di. `PetStore` È possibile assegnare un percorso base a`MyDogList`. `Dogs` API L'URLof `https://api.example.com/MyDogList` è quindi la radice URL di `Dogs`API.

Per configurare le API mappature su più livelli, puoi utilizzare solo un nome di dominio personalizzato regionale. I nomi di dominio personalizzati ottimizzati per Edge non sono supportati. Per ulteriori informazioni, consulta the section called “APImappature”.

La procedura seguente configura le API mappature per mappare i percorsi dal nome di dominio personalizzato agli stage. API

AWS Management Console

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com / apigateway.
2. Scegli Nomi di dominio personalizzati dal pannello di navigazione principale della console API Gateway.
3. Scegliere un nome di dominio personalizzato.
4. Scegli Configura API mappature.
5. Scegliere Add new mapping (Aggiungi nuova mappatura).
6. Specificate APIlo stage e il percorso (opzionale) per la mappatura.

7. Seleziona Salva.

REST API

Richiamare basepathmapping:create su un nome di dominio personalizzato specifico, indicando le proprietà `basePath`, `restApiId` e `stage` della distribuzione nel payload della richiesta.

La API chiamata riuscita restituisce una `201 Created` risposta.

Per esempi di codice della REST API chiamata, vedete basepathmapping:create.

Rotazione di un certificato importato in ACM

ACMgestisce automaticamente il rinnovo dei certificati che emette. Non è necessario ruotare i certificati ACM emessi per i nomi di dominio personalizzati. CloudFront lo gestisce per tuo conto.

Tuttavia, se importi un certificato ACM e lo utilizzi per un nome di dominio personalizzato, devi ruotare il certificato prima che scada. A tale scopo, è necessario importare un nuovo certificato di terze parti per il nome di dominio e ruotare il certificato esistente in quello nuovo. Il processo deve essere ripetuto quando il nuovo certificato importato scade. In alternativa, puoi richiedere l'emissione ACM di un nuovo certificato per il nome di dominio e ruotare quello esistente con il nuovo ACM certificato emesso. Dopodiché, puoi andartene ACM e CloudFront gestire automaticamente la rotazione dei certificati. Per creare o importare un nuovo ACM certificato, procedi nel seguente modothe section called “Per creare o importare un certificato/in SSL TLS ACM”.

La procedura seguente descrive come ruotare un certificato per un nome di dominio.

> **Note**
>
> Sono necessari circa 40 minuti per ruotare un certificato importato in. ACM

AWS Management Console

1. Richiedi o importa un certificato inACM.
2. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
3. Scegli Nomi di dominio personalizzati dal pannello di navigazione principale della console API Gateway.

4. Scegliere un nome di dominio personalizzato.
5. Seleziona Edit (Modifica).
6. Scegli il certificato desiderato dall'elenco a discesa dei ACMcertificati.
7. Scegliere Save (Salva) per iniziare la rotazione del certificato per il nome di dominio personalizzato.
8. Al termine della rotazione, puoi scegliere l'icona a forma di freccia bidirezionale accanto a ACMCertificato per tornare al certificato originale.

REST API

Chiama domainname:update action, specificando il nuovo ACM certificato per il nome ARN di dominio specificato.

Chiama il tuo con nomi di dominio personalizzati API

Chiamare un utente API con un nome di dominio personalizzato equivale a chiamarlo API con il suo nome di dominio predefinito, a condizione che URL venga utilizzato quello corretto.

Gli esempi seguenti confrontano e contrappongono un insieme di valori predefiniti URLs e personalizzati corrispondenti URLs di due APIs (`udxjefeqf3duz`) in una regione specificata (`us-east-1`) e di un determinato nome di dominio personalizzato (`api.example.com`).

APIID	Stage	Predefinito URL	Percorso di base	Personalizzato URL
udxjef	prod	https://udxjef.execute-api.us-east-1.amazonaws.com/prodotto	/petstore	https://api.example.com/negozio di animali
udxjef	tst	https://udxjef.execute-api.us-east-1.amazonaws.com/tst	/petdepot	https://api.example.com/petdepot

APIID	Stage	Predefinito URL	Percorso di base	Personalizzato URL
qf3duz	dev	https://qf3duz.execute-api.us-east-1.amazonaws.com/dev	/bookstore	https://api.example.com/libreria
qf3duz	tst	https://qf3duz.execute-api.us-east-1.amazonaws.com/tst	/bookstand	https://api.example.com/leggio

APIGateway supporta nomi di dominio personalizzati per e API utilizzando Server Name Indication () SNI. È possibile richiamarli API con un nome di dominio personalizzato utilizzando un browser o una libreria client che supportiSNI.

APIGateway si applica SNI alla distribuzione. CloudFront Per informazioni su come vengono CloudFront utilizzati i nomi di dominio personalizzati, consulta Amazon CloudFront Custom SSL.

Esegui la migrazione di un nome di dominio personalizzato a un tipo di API endpoint diverso in Gateway API

Puoi eseguire la migrazione di un nome di dominio personalizzato tra endpoint ottimizzati per edge ed endpoint regionali. Aggiungi prima di tutto il nuovo tipo di configurazione dell'endpoint all'elenco `endpointConfiguration.types` esistente per il nome di dominio personalizzato. Successivamente, si configura un DNS record per indirizzare il nome di dominio personalizzato all'endpoint appena fornito. Infine, rimuovi i dati obsoleti di configurazione del nome di dominio personalizzato.

Considerazioni

Di seguito sono riportate alcune considerazioni sulla migrazione del dominio personalizzato tra un endpoint regionale e un API endpoint ottimizzato per i dispositivi periferici. API

- Un nome di dominio personalizzato ottimizzato per i dispositivi edge richiede un certificato fornito ACM dalla regione Stati Uniti orientali (Virginia settentrionale). `us-east-1` Questo certificato è distribuito un tutte le posizioni geografiche.
- Un nome di dominio personalizzato regionale richiede un certificato fornito dalla stessa ACM regione che ospita il. API È possibile migrare un nome di dominio personalizzato ottimizzato per i dispositivi periferici che non si trova nella `us-east-1` regione verso un nome di dominio personalizzato regionale richiedendo un nuovo ACM certificato dalla regione locale a. API
- Potrebbero essere necessari fino a 60 secondi per completare una migrazione tra un nome di dominio personalizzato ottimizzato per i dispositivi periferici e un nome di dominio personalizzato regionale. Il tempo di migrazione dipende anche da quando si aggiornano i record. DNS

Esegui la migrazione di nomi di dominio personalizzati

La procedura seguente mostra come migrare un nome di dominio personalizzato ottimizzato per i dispositivi perimetrali verso un nome di dominio personalizzato regionale.

AWS Management Console

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel riquadro di navigazione principale, scegli Nomi di dominio personalizzati.
3. Scegli un nome di dominio personalizzato ottimizzato per i dispositivi periferici.
4. Nella scheda Configurazioni, per la configurazione degli endpoint, scegli Modifica.
5. Scegli Aggiungi endpoint regionale.
6. Per il ACMcertificato, scegli un certificato.

   Il certificato regionale deve essere della stessa regione del regionaleAPI.
7. Selezionare Salva.
8. Imposta un DNS record per indirizzare il nome di dominio personalizzato regionale a questo nome host regionale. Per ulteriori informazioni, vedere Configurazione di Route 53 per instradare il traffico verso API il Gateway.
9. Dopo aver confermato che la DNS configurazione utilizza l'endpoint corretto, elimini la configurazione dell'endpoint ottimizzata per i bordi. Scegli il tuo nome di dominio personalizzato e, per la configurazione degli endpoint (ottimizzata per i bordi), scegli Elimina.
10. Conferma la tua scelta ed elimina l'endpoint.

AWS CLI

Il update-domain-namecomando seguente mostra come migrare il nome di dominio personalizzato a un nome di dominio personalizzato regionale:

```
aws apigateway update-domain-name \
    --domain-name 'api.example.com' \
    --patch-operations  '[
        { "op":"add", "path": "/endpointConfiguration/types","value": "REGIONAL" },
        { "op":"add", "path": "/regionalCertificateArn", "value": "arn:aws:acm:us-
west-2:123456789012:certificate/cd833b28-58d2-407e-83e9-dce3fd852149" }
      ]'
```

Il certificato regionale deve appartenere alla stessa regione del certificato regionaleAPI.

La risposta in caso di esito positivo ha il codice di stato `200 OK` e un corpo simile al seguente:

```
{
    "certificateArn": "arn:aws:acm:us-
east-1:123456789012:certificate/34a95aa1-77fa-427c-aa07-3a88bd9f3c0a",
    "certificateName": "edge-cert",
    "certificateUploadDate": "2017-10-16T23:22:57Z",
    "distributionDomainName": "d1frvgze7vy1bf.cloudfront.net",
    "domainName": "api.example.com",
    "endpointConfiguration": {
        "types": [
            "EDGE",
            "REGIONAL"
        ]
    },
    "regionalCertificateArn": "arn:aws:acm:us-west-2:123456789012:certificate/
cd833b28-58d2-407e-83e9-dce3fd852149",
    "regionalDomainName": "d-fdisjghyn6.execute-api.us-west-2.amazonaws.com"
}
```

Per il nome di dominio personalizzato regionale migrato, la `regionalDomainName` proprietà risultante restituisce il nome API host regionale. È necessario impostare un DNS record per indirizzare il nome di dominio personalizzato regionale a questo nome host regionale. Questo consente al traffico destinato al nome di dominio personalizzato di essere instradato all'host regionale.

Dopo aver impostato il DNS record, puoi rimuovere il nome di dominio personalizzato ottimizzato per Edge. Il update-domain-namecomando seguente mostra come rimuovere il nome di dominio personalizzato ottimizzato per i bordi.

```
aws apigateway update-domain-name \
    --domain-name api.example.com \
    --patch-operations '[
            {"op":"remove", "path":"/endpointConfiguration/types", "value":"EDGE"},
            {"op":"remove", "path":"certificateName"},
            {"op":"remove", "path":"certificateArn"}
        ]'
```

La procedura seguente mostra come migrare un nome di dominio personalizzato regionale verso un nome di dominio personalizzato ottimizzato per i dispositivi periferici.

AWS Management Console

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel riquadro di navigazione principale, scegli Nomi di dominio personalizzati.
3. Scegli un nome di dominio personalizzato ottimizzato per i dispositivi periferici.
4. Nella scheda Configurazioni, per la configurazione degli endpoint, scegli Modifica.
5. Scegli Aggiungi endpoint ottimizzato per i bordi.
6. Per il ACMcertificato, scegli un certificato.

   Il certificato di dominio ottimizzato per edge deve essere creato nella regione `us-east-1`.
7. Selezionare Salva.
8. Configura un DNS record per indirizzare il nome di dominio personalizzato ottimizzato per i bordi verso questo nome host ottimizzato per i bordi. Per ulteriori informazioni, consulta la sezione Configurazione di Route 53 per indirizzare il traffico verso Gateway. API
9. Dopo aver confermato che la DNS configurazione utilizza l'endpoint corretto, elimini la configurazione regionale dell'endpoint. Scegli il tuo nome di dominio personalizzato e, per la configurazione dell'endpoint (regionale), scegli Elimina.
10. Conferma la tua scelta ed elimina l'endpoint.

AWS CLI

Il update-domain-namecomando seguente mostra come migrare il nome di dominio personalizzato verso un nome di dominio personalizzato ottimizzato per l'edge:

```
aws apigateway update-domain-name \
    --domain-name 'api.example.com' \
    --patch-operations  '[
        { "op":"add", "path": "/endpointConfiguration/types","value": "EDGE" },
        { "op":"add", "path": "/certificateName", "value": "edge-cert" },
 {"op":"add", "path": "/certificateArn", "value": "arn:aws:acm:us-
east-1:738575810317:certificate/34a95aa1-77fa-427c-aa07-3a88bd9f3c0a"}
      ]'
```

Il certificato di dominio ottimizzato per edge deve essere creato nella regione `us-east-1`.

La risposta in caso di esito positivo ha il codice di stato `200 OK` e un corpo simile al seguente:

```
{
    "certificateArn": "arn:aws:acm:us-
east-1:738575810317:certificate/34a95aa1-77fa-427c-aa07-3a88bd9f3c0a",
    "certificateName": "edge-cert",
    "certificateUploadDate": "2017-10-16T23:22:57Z",
    "distributionDomainName": "d1frvgze7vy1bf.cloudfront.net",
    "domainName": "api.example.com",
    "endpointConfiguration": {
        "types": [
            "EDGE",
            "REGIONAL"
        ]
    },
    "regionalCertificateArn": "arn:aws:acm:us-
east-1:123456789012:certificate/3d881b54-851a-478a-a887-f6502760461d",
    "regionalDomainName": "d-cgkq2qwgzf.execute-api.us-east-1.amazonaws.com"
}
```

Per il nome di dominio personalizzato specificato, API Gateway restituisce il nome host ottimizzato per i bordi come valore della proprietàAPI. `distributionDomainName` È necessario impostare un DNS record per indirizzare il nome di dominio personalizzato ottimizzato per i bordi verso questo nome di dominio di distribuzione. Ciò consente al traffico associato al nome di dominio personalizzato ottimizzato per i bordi di essere indirizzato al nome host ottimizzato per i bordi. API

Dopo aver impostato il DNS record, puoi rimuovere il tipo di endpoint del `REGION` nome di dominio personalizzato. Il `update-domain-name`comando seguente mostra come rimuovere il tipo di endpoint regionale:

```
aws apigateway update-domain-name \
    --domain-name api.example.com \
    --patch-operations '[
        {"op":"remove", "path":"/endpointConfiguration/types", value:"REGIONAL"},
        {"op":"remove", "path":"regionalCertificateArn"}
      ]'
```

Il risultato di questo comando è simile all'output seguente, con solo i dati di configurazione del nome di dominio ottimizzato per i confini:

```
{
    "certificateArn": "arn:aws:acm:us-
east-1:738575810317:certificate/34a95aa1-77fa-427c-aa07-3a88bd9f3c0a",
    "certificateName": "edge-cert",
    "certificateUploadDate": "2017-10-16T23:22:57Z",
    "distributionDomainName": "d1frvgze7vy1bf.cloudfront.net",
    "domainName": "regional.haymuto.com",
    "endpointConfiguration": {
        "types": "EDGE"
    }
}
```

Mappa API le fasi su un nome di dominio personalizzato per REST APIs

Le API mappature vengono utilizzate per connettere le API fasi a un nome di dominio personalizzato. Dopo aver creato un nome di dominio e configurato DNS i record, utilizzi le API mappature per inviare traffico al tuo utente APIs tramite il tuo nome di dominio personalizzato.

Una API mappatura specifica unaAPI, una fase e, facoltativamente, un percorso da utilizzare per la mappatura. Ad esempio, è possibile mappare lo `production` stadio di un albero. API `https://api.example.com/orders`

È possibile mappare HTTP tutti gli REST API stage sullo stesso nome di dominio personalizzato.

Prima di creare una API mappatura, è necessario disporre di unaAPI, di una fase e di un nome di dominio personalizzato. Per ulteriori informazioni sulla creazione di un nome di dominio personalizzato, consulta the section called “Configura un nome di dominio personalizzato regionale”.

Richieste di routing API

È possibile configurare API mappature con più livelli, ad esempio e. `orders/v1/items` `orders/v2/items`

> **Note**
>
> Per configurare API mappature con più livelli, è necessario utilizzare un nome di dominio personalizzato regionale con la TLS politica di sicurezza 1.2.

Per le API mappature con più livelli, API Gateway indirizza le richieste alla API mappatura con il percorso corrispondente più lungo. APIGateway considera solo i percorsi configurati per le API mappature, e non i API percorsi, per selezionare quelli da richiamare. API Se nessun percorso corrisponde alla richiesta, API Gateway invia la richiesta a API quello che hai mappato sul percorso vuoto. `(none)`

Per i nomi di dominio personalizzati che utilizzano API mappature con più livelli, API Gateway indirizza le richieste alla API mappatura con il prefisso corrispondente più lungo.

Ad esempio, considera un nome `https://api.example.com` di dominio personalizzato con le seguenti mappature: API

1. `(none)`mappato su 1. API
2. `orders`mappato su 2API.
3. `orders/v1/items`mappato a 3. API
4. `orders/v2/items`mappato a 4. API
5. `orders/v2/items/categories`mappato a 5. API

Richiesta	Selezionato API	Spiegazione
`https://api.example.com/orders`	`API 2`	La richiesta corrisponde esattamente a questa API mappatura.
`https://api.example.com/orders/v1/items`	`API 3`	La richiesta corrisponde esattamente a questa API mappatura.
`https://api.example.com/orders/v2/items`	`API 4`	La richiesta corrisponde esattamente a questa API mappatura.
`https://api.example.com/orders/v1/items/123`	`API 3`	APIGateway sceglie la mappatura con il percorso corrispondente più lungo. `123` alla fine della richiesta non influisce sulla selezione.
`https://api.example.com/orders/v2/items/categories/5`	`API 5`	APIGateway sceglie la mappatura con il percorso corrispondente più lungo.
`https://api.example.com/customers`	`API 1`	APIGateway utilizza la mappatura vuota come soluzione generale.
`https://api.example.com/ordersandmore`	`API 2`	APIGateway sceglie la mappatura con il prefisso corrispondente più lungo. Per un nome di dominio personalizzato configurato con mappature a livello singolo, ad esempio only `https://api.example.com/orders` and, API Gateway scegliere bbe `https://a`

Richiesta	Selezionato API	Spiegazione
		`pi.example.com/` , poiché non esiste un percorso `API 1` corrispondente con. `ordersandmore`

Restrizioni

- In una API mappatura, il nome di dominio personalizzato e quello mappato APIs devono trovarsi nello stesso account. AWS
- APIle mappature devono contenere solo lettere, numeri e i seguenti caratteri:. `$-_.+!*'()/`
- La lunghezza massima del percorso in una API mappatura è di 300 caratteri.
- Puoi avere 200 API mappature con più livelli per ogni nome di dominio.
- È possibile eseguire il mapping HTTP APIs solo su un nome di dominio personalizzato regionale con la politica di sicurezza TLS 1.2.
- Non è possibile eseguire il mapping WebSocket APIs allo stesso nome di dominio personalizzato di un HTTP API or RESTAPI.

Creare una API mappatura

Per creare una API mappatura, devi prima creare un nome di dominio e uno stage API personalizzati. Per informazioni sulla creazione di un nome di dominio personalizzato, consulta the section called "Configura un nome di dominio personalizzato regionale".

Per esempio i AWS Serverless Application Model modelli che creano tutte le risorse, vedi Sessions With SAM on GitHub.

AWS Management Console

Per creare una API mappatura

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere Nomi di dominio personalizzati.
3. Selezionare un nome di dominio personalizzato già creato.
4. Scegli le APImappature.

5. Scegli Configura mappature API.
6. Scegliere Add new mapping (Aggiungi nuova mappatura).
7. Inserisci uno API, uno stadio e, facoltativamente, un percorso.
8. Selezionare Salva.

AWS CLI

Il AWS CLI comando seguente crea una API mappatura. In questo esempio, API Gateway invia le richieste `api.example.com/v1/orders` allo stadio API e specificato.

> **Note**
>
> Per creare API mappature con più livelli, è necessario utilizzare. `apigatewayv2`

```
aws apigatewayv2 create-api-mapping \
    --domain-name api.example.com \
    --api-mapping-key v1/orders \
    --api-id a1b2c3d4 \
    --stage test
```

AWS CloudFormation

L' AWS CloudFormation esempio seguente crea una API mappatura.

> **Note**
>
> Per creare API mappature con più livelli, è necessario utilizzare. `AWS::ApiGatewayV2`

```
MyApiMapping:
  Type: 'AWS::ApiGatewayV2::ApiMapping'
  Properties:
    DomainName: api.example.com
    ApiMappingKey: 'orders/v2/items'
    ApiId: !Ref MyApi
    Stage: !Ref MyStage
```

Scegli una politica di sicurezza per il tuo dominio REST API personalizzato in API Gateway

Per una maggiore sicurezza del tuo dominio personalizzato Amazon API Gateway, puoi scegliere una policy di sicurezza nella console API Gateway AWS CLI, o un AWS SDK.

Una policy di sicurezza è una combinazione predefinita di TLS versioni minime e suite di crittografia offerte da Gateway. API È possibile scegliere una politica di sicurezza della TLS versione 1.2 o della TLS versione 1.0. Il TLS protocollo risolve problemi di sicurezza della rete come la manomissione e l'intercettazione tra client e server. Quando i client stabiliscono una TLS stretta di mano nei vostri confronti API tramite il dominio personalizzato, la politica di sicurezza applica la TLS versione e le opzioni della suite di crittografia che i clienti possono scegliere di utilizzare.

Nelle impostazioni del dominio personalizzato, una policy di sicurezza determina due impostazioni:

- La TLS versione minima utilizzata da API Gateway per comunicare con i client API
- Il codice utilizzato da API Gateway per crittografare il contenuto che restituisce ai client API

Se scegli una politica di sicurezza TLS 1.0, la politica di sicurezza accetta il traffico TLS 1.0, TLS 1.2 e TLS 1.3. Se scegli una politica di sicurezza TLS 1.2, la politica di sicurezza accetta il traffico TLS 1.2 e TLS 1.3 e rifiuta il traffico TLS 1.0.

> **Note**
>
> È possibile specificare una politica di sicurezza solo per un dominio personalizzato. Per API utilizzare un endpoint predefinito, API Gateway utilizza la seguente politica di sicurezza:
>
> - Per soluzioni ottimizzate per i dispositivi periferici: APIs `TLS-1-0`
> - Per Regional: APIs `TLS-1-0`
> - Per uso privatoAPIs: `TLS-1-2`
>
> I codici per ogni politica di sicurezza sono descritti nelle tabelle seguenti di questa pagina.

Argomenti

- Come specificare una politica di sicurezza per i domini personalizzati

- Politiche di sicurezza, versioni di TLS protocollo e cifrari supportati per domini personalizzati ottimizzati per i dispositivi perimetrali
- Politiche di sicurezza, versioni di TLS protocollo e cifrari supportati per domini personalizzati regionali
- Versioni di TLS protocollo e cifrari supportati per uso privato APIs
- Nomi aperti SSL e cifrati RFC
- Informazioni su HTTP APIs e WebSocket APIs

Come specificare una politica di sicurezza per i domini personalizzati

Quando si crea un nome di dominio personalizzato, si specifica la relativa politica di sicurezza. Per informazioni su come creare un dominio personalizzato, consulta the section called “Configura un nome di dominio personalizzato ottimizzato per i dispositivi periferici in Gateway API” othe section called “Configura un nome di dominio personalizzato regionale”.

Per modificare la politica di sicurezza del tuo nome di dominio personalizzato, aggiorna le impostazioni del dominio personalizzato. Puoi aggiornare le impostazioni del nome di dominio personalizzato utilizzando il AWS Management Console AWS CLI, o un AWS SDK.

Quando si utilizza il API Gateway REST API o AWS CLI, si specifica la nuova TLS versione `TLS_1_0` o `TLS_1_2` nel `securityPolicy` parametro. Per ulteriori informazioni, consulta domainname:update in Amazon API Gateway REST API Reference o update-domain-namenel Reference.AWS CLI

Il completamento dell'operazione di aggiornamento potrebbe richiedere alcuni minuti.

Politiche di sicurezza, versioni di TLS protocollo e cifrari supportati per domini personalizzati ottimizzati per i dispositivi perimetrali

La tabella seguente descrive le politiche di sicurezza che possono essere specificate per i nomi di dominio personalizzati ottimizzati per i bordi.

TLSprotocolli	TLSpolitica di sicurezza _1_0	TLS_1_2 politica di sicurezza
TLSv1.3.	Sì	Sì

TLSprotocolli	TLSpolitica di sicurezza _1_0	TLS_1_2 politica di sicurezza
TLSv12.	Sì	Sì
TLSv1.1.	Sì	No
TLSv1	Sì	No

La tabella seguente descrive i TLS codici disponibili per ogni politica di sicurezza.

TLScifrari	TLSpolitica di sicurezza _1_0	TLS_1_2 politica di sicurezza
TLS_ _128_ _ AES GCM SHA256	Sì	Sì
TLS_ _256_ _ AES GCM SHA384	Sì	Sì
TLS_ 0_ 05_ CHACHA2 POLY13 SHA256	Sì	Sì
ECDHE-ECDSA-AES128-GCM-SHA256	Sì	Sì
ECDHE-ECDSA-AES128-SHA256	Sì	Sì
ECDHE-ECDSA-AES128-SHA	Sì	No
ECDHE-ECDSA-AES256-GCM-SHA384	Sì	Sì
ECDHE- ECDSA - CHACHA2 0- 05 POLY13	Sì	Sì

TLScifrari	TLSpolitica di sicurezza _1_0	TLS_1_2 politica di sicurezza
ECDHE-ECDSA-AES256-SHA384	Sì	Sì
ECDHE-ECDSA-AES256-SHA	Sì	No
ECDHE-RSA-AES128-GCM-SHA256	Sì	Sì
ECDHE-RSA-AES128-S HA256	Sì	Sì
ECDHE-RSA-AES128-SHA	Sì	No
ECDHE-RSA-AES256-GCM-SHA384	Sì	Sì
ECDHE- RSA - CHACHA2 0-POLY13 05	Sì	Sì
ECDHE-RSA-AES256-S HA384	Sì	Sì
ECDHE-RSA-AES256-SHA	Sì	No
AES128-GCM-SHA256	Sì	No
AES256-GCM-SHA384	Sì	Sì
AES128-SHA256	Sì	Sì
AES256-SHA	Sì	No

TLScifrari	TLSpolitica di sicurezza _1_0	TLS_1_2 politica di sicurezza
AES128-SHA	Sì	No
DES-CBC3-SHA	Sì	No

Politiche di sicurezza, versioni di TLS protocollo e cifrari supportati per domini personalizzati regionali

La tabella seguente descrive le politiche di sicurezza per i nomi di dominio personalizzati regionali.

TLSprotocolli	TLSpolitica di sicurezza _1_0	TLS_1_2 politica di sicurezza
TLSv1.3.	Sì	Sì
TLSv12.	Sì	Sì
TLSv1.1.	Sì	No
TLSv1	Sì	No

La tabella seguente descrive i TLS codici disponibili per ogni politica di sicurezza.

TLScifrari	TLSpolitica di sicurezza _1_0	TLS_1_2 politica di sicurezza
TLS_ _128_ _ AES GCM SHA256	Sì	Sì
TLS_ _256_ _ AES GCM SHA384	Sì	Sì

TLScifrari	TLSpolitica di sicurezza _1_0	TLS_1_2 politica di sicurezza
TLS_ 0_ 05_ CHACHA2 POLY13 SHA256	Sì	Sì
ECDHE-ECDSA-AES128-GCM-SHA256	Sì	Sì
ECDHE-RSA-AES128-GCM-SHA256	Sì	Sì
ECDHE-ECDSA-AES128-SHA256	Sì	Sì
ECDHE-RSA-AES128-S HA256	Sì	Sì
ECDHE-ECDSA-AES128-SHA	Sì	No
ECDHE-RSA-AES128-SHA	Sì	No
ECDHE-ECDSA-AES256-GCM-SHA384	Sì	Sì
ECDHE-RSA-AES256-GCM-SHA384	Sì	Sì
ECDHE-ECDSA-AES256-SHA384	Sì	Sì
ECDHE-RSA-AES256-S HA384	Sì	Sì
ECDHE-RSA-AES256-SHA	Sì	No
ECDHE-ECDSA-AES256-SHA	Sì	No

TLScifrari	TLSpolitica di sicurezza _1_0	TLS_1_2 politica di sicurezza
AES128-GCM-SHA256	Sì	Sì
AES128-SHA256	Sì	Sì
AES128-SHA	Sì	No
AES256-GCM-SHA384	Sì	Sì
AES256-SHA256	Sì	Sì
AES256-SHA	Sì	No

Versioni di TLS protocollo e cifrari supportati per uso privato APIs

La tabella seguente descrive i TLS protocolli supportati per uso privato. APIs La specificazione di una politica di sicurezza per uso privato non APIs è supportata.

TLSprotocolli	TLS_1_2 politica di sicurezza
TLSv1.2.	Sì

La tabella seguente descrive TLS i codici disponibili per la politica di `TLS_1_2` sicurezza privataAPIs. ogni politica di sicurezza.

TLScifrari	TLS_1_2 politica di sicurezza
ECDHE-ECDSA-AES128-GCM-SHA256	Sì
ECDHE-RSA-AES128-GCM-SHA256	Sì
ECDHE-ECDSA-AES128-SHA256	Sì
ECDHE-RSA-AES128-SHA256	Sì
ECDHE-ECDSA-AES256-GCM-SHA384	Sì
ECDHE-RSA-AES256-GCM-SHA384	Sì
ECDHE-ECDSA-AES256-SHA384	Sì
ECDHE-RSA-AES256-SHA384	Sì
AES128-GCM-SHA256	Sì
AES128-SHA256	Sì
AES256-GCM-SHA384	Sì
AES256-SHA256	Sì

Nomi aperti SSL e cifrati RFC

Open SSL e IETF RFC 5246 utilizzano nomi diversi per gli stessi cifrari. La tabella seguente associa il SSL nome Open al nome di ogni RFC cifrario.

Nome cifrato aperto SSL	RFCnome del cifrario
TLS_ _128_ _ AES GCM SHA256	TLS_ _128_ _ AES GCM SHA256
TLS_ _256_ _ AES GCM SHA384	TLS_ _256_ _ AES GCM SHA384
TLS_ 0_ 05_ CHACHA2 POLY13 SHA256	TLS_ 0_ 05_ CHACHA2 POLY13 SHA256
ECDHE-RSA-AES128-GCM-SHA256	TLS_ _ _ ECDHE _ RSA _128_ WITH _ AES GCM SHA256
ECDHE-RSA-AES128-SHA256	TLS_ _ _ ECDHE _ RSA _128_ WITH _ AES CBC SHA256
ECDHE-RSA-AES128-SHA	TLS_ _ _ ECDHE _ RSA _128_ WITH _ AES CBC SHA
ECDHE-RSA-AES256-GCM-SHA384	TLS_ _ _ ECDHE _ RSA _256_ WITH _ AES GCM SHA384
ECDHE-RSA-AES256-SHA384	TLS_ _ _ ECDHE _ RSA _256_ WITH _ AES CBC SHA384
ECDHE-RSA-AES256-SHA	TLS_ _ _ ECDHE _ RSA _256_ WITH _ AES CBC SHA

Nome cifrato aperto SSL	RFCnome del cifrario
AES128-GCM-SHA256	TLS_ _ _ RSA _128_ WITH _ AES GCM SHA256
AES256-GCM-SHA384	TLS_ _ _ RSA _256_ WITH _ AES GCM SHA384
AES128-SHA256	TLS_ _ _ RSA _128_ WITH _ AES CBC SHA256
AES256-SHA	TLS_ _ _ RSA _256_ WITH _ AES CBC SHA
AES128-SHA	TLS_ _ _ RSA _128_ WITH _ AES CBC SHA
DES-CBC3-SHA	TLS_ RSA _ WITH _3 DES _ _ _ EDE CBC SHA

Informazioni su HTTP APIs e WebSocket APIs

Per ulteriori informazioni su HTTP APIs e WebSocket APIs, vedere the section called “Politica di sicurezza per HTTP APIs in API Gateway” ethe section called “Politica di sicurezza per WebSocket APIs in API Gateway”.

Disabilita l'endpoint predefinito per le API REST

Per impostazione predefinita, i client possono richiamare l'API utilizzando l'endpoint `execute-api` generato da API Gateway per l'API. Per garantire che i client possano accedere all'API solo utilizzando un nome di dominio personalizzato con l'autenticazione TLS reciproca, disattivare l'endpoint `execute-api` predefinito. I client possono comunque connettersi all'endpoint predefinito, ma riceveranno un codice di stato `403 Forbidden`. Quando si disattiva l'endpoint predefinito, questa operazione influisce su tutte le fasi di un'API.

La procedura seguente mostra come disabilitare l'endpoint predefinito per un'API REST.

AWS Management Console

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere una REST API.
3. Nel riquadro di navigazione principale, scegli Impostazioni API.

4. Scegliere un'API.
5. Nei dettagli dell'API, scegli Modifica.
6. Per Endpoint predefinito, seleziona Inattivo.
7. Seleziona Salvataggio delle modifiche.
8. Nel riquadro di navigazione principale, scegli Risorse.
9. Seleziona Deploy API (Distribuisci API).
10. Ridistribuisci l'API o crea una nuova fase per rendere effettiva la modifica.

AWS CLI

Il update-rest-apicomando seguente mostra come disabilitare l'endpoint predefinito:

```
aws apigateway update-rest-api \
    --rest-api-id abcdef123 \
    --patch-operations op=replace,path=/disableExecuteApiEndpoint,value='True'
```

Dopo aver disabilitato l'endpoint predefinito, è necessario distribuire l'API per rendere effettiva la modifica.

Il seguente comando create-deployment crea una distribuzione:

```
aws apigateway create-deployment \
    --rest-api-id abcdef123 \
    --stage-name dev
```

Configurazione di controlli di integrità personalizzati per il DNS failover di un gateway API API

Puoi utilizzare i controlli di integrità di Amazon Route 53 per controllare il DNS failover da un API gateway API in una regione primaria Regione AWS a uno in una regione secondaria. Questo può aiutare a mitigare gli impatti in caso di problema regionale. Se utilizzi un dominio personalizzato, puoi eseguire il failover senza richiedere ai client di modificare API gli endpoint.

Quando si sceglie Evaluate Target Health per un record di alias, tali record hanno esito negativo solo quando il servizio API Gateway non è disponibile nella regione. In alcuni casi, il tuo API Gateway

APIs può subire interruzioni prima di quel momento. Per controllare direttamente il DNS failover, configura i controlli di integrità personalizzati di Route 53 per il tuo API gateway. APIs Per questo esempio, si utilizza un CloudWatch allarme che aiuta gli operatori a controllare il DNS failover. Per ulteriori esempi e altre considerazioni sulla configurazione del failover, vedere Creazione di meccanismi di disaster recovery utilizzando Route 53 ed Esecuzione dei controlli di integrità di Route 53 su risorse private in un VPC unico ambiente. AWS Lambda CloudWatch

Argomenti

- Prerequisiti
- Fase 1: Configurazione delle risorse
- Fase 2: Avvio del failover nella regione secondaria
- Fase 3: Test del failover
- Fase 4: Ritorno alla regione primaria
- Fasi successive: personalizzare e verificare periodicamente

Prerequisiti

Per completare questa procedura, è necessario creare e configurare le risorse seguenti:

- Un nome di dominio di cui si è proprietari.
- Un ACM certificato per quel nome di dominio in due. Regioni AWS Per ulteriori informazioni, consulta the section called “Prerequisiti ”.
- Una zona ospitata Route 53 per il nome di dominio in uso. Per ulteriori informazioni, consulta Utilizzo delle zone ospitate nella Guida per gli sviluppatori di Amazon Route 53.

Per ulteriori informazioni su come creare DNS record di failover Route 53 per i nomi di dominio, consulta Choose a routing policy nella Amazon Route 53 Developer Guide. Per ulteriori informazioni su come monitorare un CloudWatch allarme, consulta Monitoring a CloudWatch alarm nella Amazon Route 53 Developer Guide.

Fase 1: Configurazione delle risorse

In questo esempio, crei le seguenti risorse per configurare il DNS failover per il tuo nome di dominio:

- APIGateway APIs in due Regioni AWS
- APINomi di dominio personalizzati Gateway con lo stesso nome in due Regioni AWS

- APIAPIMappature del gateway che collegano il API gateway APIs ai nomi di dominio personalizzati
- DNSRecord di failover Route 53 per i nomi di dominio
- Un CloudWatch allarme nella regione secondaria
- Un controllo dello stato della Route 53 basato sull' CloudWatch allarme nella regione secondaria

Innanzitutto, accertarsi di disporre di tutte le risorse richieste nelle regioni primaria e secondaria. La regione secondaria deve contenere l'allarme e il controllo dell'integrità. In questo modo, non si dipende dalla regione primaria per eseguire il failover. Per esempio AWS CloudFormation i modelli che creano queste risorse, vedi `primary.yaml`e `secondary.yaml`.

> **Important**
>
> Prima del failover nella regione secondaria, accertarsi che tutte le risorse necessarie siano disponibili. Altrimenti, non API sarai pronto per il traffico nella regione secondaria.

Fase 2: Avvio del failover nella regione secondaria

Nell'esempio seguente, la regione di standby riceve una CloudWatch metrica e avvia il failover. Utilizziamo una metrica personalizzata che richiede l'intervento dell'operatore per avviare il failover.

```
aws cloudwatch put-metric-data \
    --metric-name Failover \
    --namespace HealthCheck \
    --unit Count \
    --value 1 \
    --region us-west-1
```

Sostituisci i dati metrici con i dati corrispondenti per l'allarme che hai configurato. CloudWatch

Fase 3: Test del failover

Invoca la tua API e verifica di ricevere una risposta dalla regione secondaria. Se si sono utilizzati i modelli di esempio della fase 1, la risposta cambia da `{"message": "Hello from the primary Region!"}` a `{"message": "Hello from the secondary Region!"}` dopo il failover.

```
curl https://my-api.example.com
```

```
{"message": "Hello from the secondary Region!"}
```

Fase 4: Ritorno alla regione primaria

Per tornare alla regione principale, invia una CloudWatch metrica che determini il superamento del controllo sanitario.

```
aws cloudwatch put-metric-data \
    --metric-name Failover \
    --namespace HealthCheck \
    --unit Count \
    --value 0 \
    --region us-west-1
```

Sostituisci i dati metrici con i dati corrispondenti per l' CloudWatch allarme che hai configurato.

Invoca la tua API e verifica di ricevere una risposta dalla regione principale. Se si sono utilizzati i modelli di esempio della fase 1, la risposta cambia da `{"message": "Hello from the secondary Region!"}` a `{"message": "Hello from the primary Region!"}`.

```
curl https://my-api.example.com

{"message": "Hello from the primary Region!"}
```

Fasi successive: personalizzare e verificare periodicamente

Questo esempio dimostra un modo per configurare DNS il failover. È possibile utilizzare una varietà di CloudWatch metriche o HTTP endpoint per i controlli di integrità che gestiscono il failover. Verificare periodicamente i meccanismi di failover per accertarsi che funzionino come previsto e che gli operatori conoscano le procedure di failover.

Ottimizzazione delle prestazioni delle API REST

Dopo aver reso disponibile la tua API per essere chiamata, potrebbe essere necessario ottimizzarla per renderla più reattività. API Gateway fornisce alcune strategie per ottimizzare l'API, come il caching delle risposte e la compressione del payload. In questa sezione viene descritto come abilitare queste funzionalità.

Argomenti

- Impostazioni della cache per REST APIs in API Gateway
- Compressione del payload per REST APIs in API Gateway

Impostazioni della cache per REST APIs in API Gateway

È possibile abilitare la API memorizzazione nella cache in API Gateway per memorizzare nella cache le risposte dell'endpoint. Con la memorizzazione nella cache, puoi ridurre il numero di chiamate effettuate all'endpoint e anche migliorare la latenza delle richieste al tuo. API

Quando abiliti la memorizzazione nella cache per una fase, API Gateway memorizza nella cache le risposte dall'endpoint per un periodo specificato time-to-live ()TTL, in secondi. APIGateway risponde quindi alla richiesta cercando la risposta dell'endpoint dalla cache invece di effettuare una richiesta all'endpoint. Il TTL valore predefinito per la API memorizzazione nella cache è 300 secondi. Il TTL valore massimo è 3600 secondi. TTL=0 significa che la memorizzazione nella cache è disabilitata.

> **Note**
>
> Il caching è il miglior tentativo. Puoi utilizzare le `CacheMissCount` metriche `CacheHitCount` e di Amazon CloudWatch per monitorare le richieste che API Gateway fornisce dalla API cache.

La dimensione massima di una risposta che può essere memorizzata nella cache è 1048576 byte. la crittografia dei dati della cache può aumentare le dimensioni della risposta quando viene memorizzata nella cache.

Questo è un servizio HIPAA idoneo. Per ulteriori informazioni sull' AWS U.S. Health Insurance Portability and Accountability Act del 1996 (HIPAA) e sull'utilizzo AWS dei servizi per elaborare, archiviare e trasmettere informazioni sanitarie protette (PHI), vedere HIPAA Panoramica.

> **Important**
>
> Quando abiliti il caching per una fase, il sistema di caching sarà abilitato per impostazione predefinita solo per i metodi `GET`. Questo aiuta a garantire la sicurezza e la disponibilità del tuo. API Puoi abilitare il caching per altri metodi ignorando le impostazioni del metodo.

> **Important**
>
> Per il caching viene applicato un addebito orario in base alle dimensioni della cache selezionata. La memorizzazione nella cache non è idonea per il livello AWS gratuito. Per ulteriori informazioni, consulta APIGateway Pricing.

Abilita la memorizzazione nella cache di Amazon API Gateway

In API Gateway, puoi abilitare la memorizzazione nella cache per una fase specifica.

Quando abiliti il caching, devi scegliere una capacità di cache. In generale, una capacità più ampia garantisce prestazioni migliori ma comporta costi più alti. Per le dimensioni di cache supportate, vedere cacheClusterSizenel APIGateway API Reference.

APIGateway consente la memorizzazione nella cache creando un'istanza di cache dedicata. Questo processo può richiedere fino a 4 minuti.

APIGateway modifica la capacità di memorizzazione nella cache rimuovendo l'istanza di cache esistente e creandone una nuova con una capacità modificata. Tutti i dati esistenti memorizzati nella cache vengono eliminati.

> **Note**
>
> La capacità della cache influisce sulla CPU memoria e sulla larghezza di banda di rete dell'istanza di cache. Di conseguenza, la capacità della cache può influire sulle prestazioni della cache.
> APIGateway consiglia di eseguire un test di carico di 10 minuti per verificare che la capacità della cache sia adeguata al carico di lavoro. Assicurati che il traffico durante il test di carico rispecchi il traffico di produzione. Ad esempio, includi un aumento graduale, un traffico costante e picchi di traffico. Il test di caricamento deve includere risposte che possono essere fornite dalla cache, nonché risposte univoche che aggiungono elementi alla cache. Durante il test di caricamento, monitora i parametri di latenza, 4xx, 5xx, hit della cache e mancato riscontro nella cache. In base a questi parametri, regola la capacità della cache a seconda delle esigenze. Per ulteriori informazioni sui test di carico, vedi Come faccio a selezionare la migliore capacità della cache del API Gateway per evitare di raggiungere un limite di velocità? .

Nella console API Gateway, è possibile configurare la memorizzazione nella cache nella pagina Stages. Si effettua il provisioning della cache dello stage e si specifica un'impostazione predefinita della cache a livello di metodo. Se attivate la cache a livello di metodo predefinita, la memorizzazione nella cache a livello di metodo viene attivata per tutti i `GET` metodi sullo stage, a meno che tale metodo non disponga di un'alternativa al metodo. Tutti `GET` i metodi aggiuntivi distribuiti sullo stage avranno una cache a livello di metodo. Per configurare l'impostazione della memorizzazione nella cache a livello di metodo per metodi specifici dello stage, puoi utilizzare le sostituzioni dei metodi. Per ulteriori informazioni sulle sostituzioni dei metodi, consulta. the section called “Come ignorare il caching di fase per il caching dei metodi”

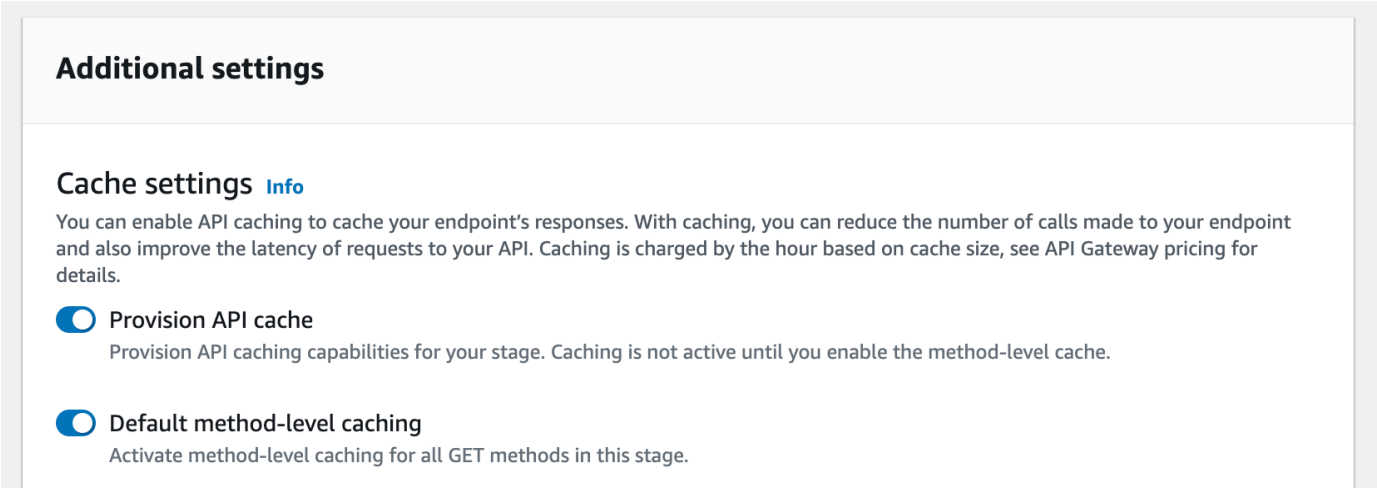
Per configurare la API memorizzazione nella cache per una determinata fase:

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere Stages (Fasi).
3. Nell'elenco delle tappe diAPI, scegli la fase.
4. Nella sezione Dettagli fase scegli Modifica.
5. In Impostazioni aggiuntive, per Impostazioni cache, attiva Provision API cache.

   Questo fornisce un cluster di cache per il tuo stage.
6. Per attivare la memorizzazione nella cache per lo stage, attiva la memorizzazione nella cache a livello di metodo predefinito.

   Questo attiva la memorizzazione nella cache a livello di metodo per tutti i metodi sullo stage. `GET` Tutti `GET` i metodi aggiuntivi distribuiti in questa fase avranno una cache a livello di metodo.

   > **Note**
   >
   > Se si dispone di un'impostazione esistente per una cache a livello di metodo, la modifica dell'impostazione predefinita della memorizzazione nella cache a livello di metodo non influisce sull'impostazione esistente.



7. Scegli Save changes (Salva modifiche).

> **Note**
>
> Il completamento della creazione o dell'eliminazione di una cache richiede circa 4 minuti per API il completamento di Gateway.
> Quando viene creata una cache, il valore del cluster di cache cambia da `Create in progress` a`Active`. Una volta completata l'eliminazione della cache, il valore del cluster di cache cambia da `Delete in progress` a`Inactive`.
> Quando attivate la memorizzazione nella cache a livello di metodo per tutti i metodi sullo stage, il valore predefinito di memorizzazione nella cache a livello di metodo cambia in. `Active` Se disattivate la memorizzazione nella cache a livello di metodo per tutti i metodi sullo stage, il valore di memorizzazione nella cache a livello di metodo predefinito cambia in. `Inactive` Se disponete di un'impostazione esistente per una cache a livello di metodo, la modifica dello stato della cache non influisce su tale impostazione.

Se abiliti il caching in Impostazioni cache di una fase, vengono memorizzati nella cache solo i metodi `GET`. Per garantire la sicurezza e la disponibilità del tuoAPI, ti consigliamo di non modificare questa impostazione. Tuttavia, puoi abilitare il caching per altri metodi ignorando le impostazioni del metodo.

Puoi verificare che il caching funzioni come previsto in due modi:

- Controlla le CloudWatch metriche di CacheHitCounte CacheMissCountper te e per StageAPI.
- Inserisci un timestamp nella risposta.

> **Note**
>
> Non dovreste usare l'`X-Cache`intestazione della CloudFront risposta per determinare se la vostra istanza di cache API Gateway API è servita.

Sostituisci la memorizzazione nella cache a livello di fase di API Gateway per la memorizzazione nella cache a livello di metodo

È possibile sovrascrivere le impostazioni della cache a livello di fase attivando o disattivando la memorizzazione nella cache per un metodo specifico. È inoltre possibile modificare il TTL periodo o attivare o disattivare la crittografia per le risposte memorizzate nella cache.

Se modificate l'impostazione predefinita della memorizzazione nella cache a livello di metodo nei dettagli dello stage, ciò non influirà sulle impostazioni della cache a livello di metodo che hanno delle sostituzioni.

Se prevedi che nelle risposte di un metodo che si sta memorizzando nella cache siano inclusi dati sensibili, in Cache Settings (Impostazioni cache), scegliere Encrypt cache data (Crittografa dati cache).

Per configurare la memorizzazione nella API cache per singoli metodi utilizzando la console:

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli il. API
3. Scegliere Stages (Fasi).
4. Nell'elenco Fasi diAPI, espandi lo stage e scegli un metodo inAPI.
5. Nella sezione Sostituzioni del metodo, scegli Modifica.
6. Nella sezione Impostazioni metodo, attiva o disattiva Abilita cache metodo o personalizza le altre opzioni desiderate.

   > **Note**
   >
   > La memorizzazione nella cache non è attiva finché non si esegue il provisioning di un cluster di cache per lo stage.

7. Seleziona Salva.

Uso dei parametri di metodo o di integrazione come chiavi di cache per indicizzare le risposte memorizzate nella cache

Quando un metodo o un'integrazione memorizzati nella cache dispone di parametri, che possono assumere la forma di intestazioni, URL percorsi o stringhe di query personalizzate, è possibile utilizzare alcuni o tutti i parametri per creare chiavi di cache. APIGateway può memorizzare nella cache le risposte del metodo, a seconda dei valori dei parametri utilizzati.

> **Note**
>
> Le chiavi di cache sono necessarie per la configurazione del caching su una risorsa.

Ad esempio, supponiamo di avere una richiesta espressa nel seguente formato:

```
GET /users?type=... HTTP/1.1
host: example.com
...
```

In questa richiesta `type` può assumere il valore `admin` o `regular`. Se includi il parametro `type` come parte della chiave di cache, le risposte da `GET /users?type=admin` vengono memorizzate nella cache separatamente da quelle di `GET /users?type=regular`.

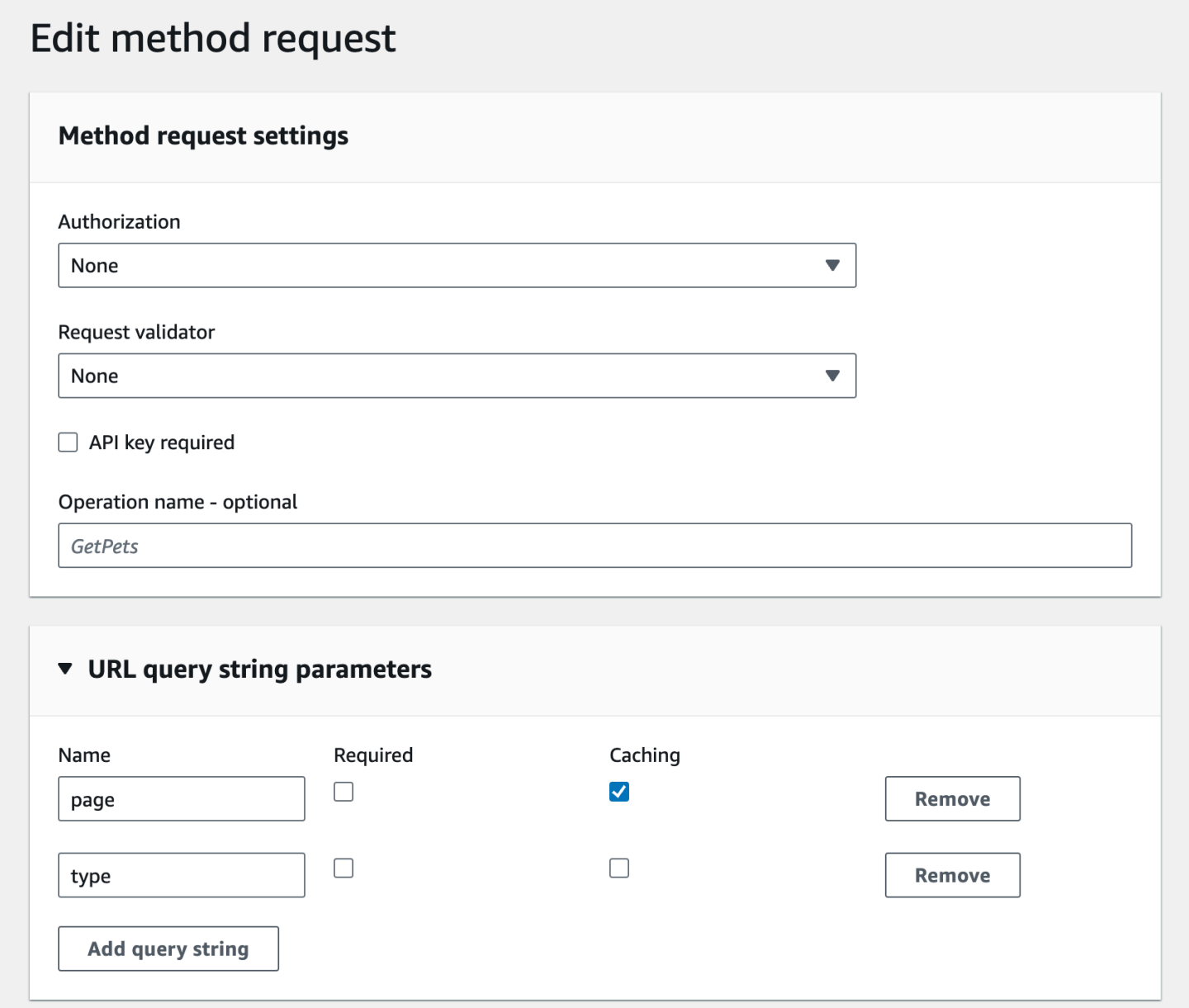
Quando una richiesta di metodo o di integrazione usa più di un parametro, puoi scegliere di includere alcuni di essi o tutti per creare la chiave di cache. Ad esempio, è possibile includere solo il `type` parametro nella chiave della cache per la seguente richiesta, effettuata nell'ordine elencato entro un TTL periodo:

```
GET /users?type=admin&department=A HTTP/1.1
host: example.com
...
```

La risposta da questa richiesta viene memorizzata nella cache e utilizzata per servire la seguente richiesta:

```
GET /users?type=admin&department=B HTTP/1.1
host: example.com
...
```

Per includere un metodo o un parametro di richiesta di integrazione come parte di una chiave di cache nella console API Gateway, seleziona Caching dopo aver aggiunto il parametro.



Svuota la cache dello API stage in Gateway API

Quando la API memorizzazione nella cache è abilitata, puoi svuotare la cache dello API stage per garantire che i API clienti ricevano le risposte più recenti dagli endpoint di integrazione.

Per svuotare la cache API dello stage, scegli il menu delle azioni dello stage, quindi seleziona Flush stage cache.

> **Note**
>
> Una volta scaricata la cache, le risposte vengono servite dall'endpoint di integrazione fino a quando la cache non viene creata nuovamente. Durante questo periodo, il numero di richieste inviate all'endpoint di integrazione potrebbe aumentare. Ciò potrebbe aumentare temporaneamente la latenza complessiva del tuo. API

Invalidare una voce della cache del Gateway API

Un vostro cliente API può invalidare una voce di cache esistente e ricaricarla dall'endpoint di integrazione per singole richieste. Il client deve inviare una richiesta contenente l'intestazione `Cache-Control: max-age=0`. Il client riceve la risposta direttamente dall'endpoint di integrazione anziché dalla cache, a condizione che il client sia autorizzato a eseguire questa operazione. In questo modo, la voce di cache esistente viene sostituita con la nuova risposta recuperata dall'endpoint di integrazione.

Per concedere l'autorizzazione a un client, allega una policy del seguente formato a un ruolo di IAM esecuzione per l'utente.

> **Note**
>
> La convalida della cache tra più account non è supportata.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:InvalidateCache"
      ],
      "Resource": [
        "arn:aws:execute-api:region:account-id:api-id/stage-name/GET/resource-path-specifier"
```

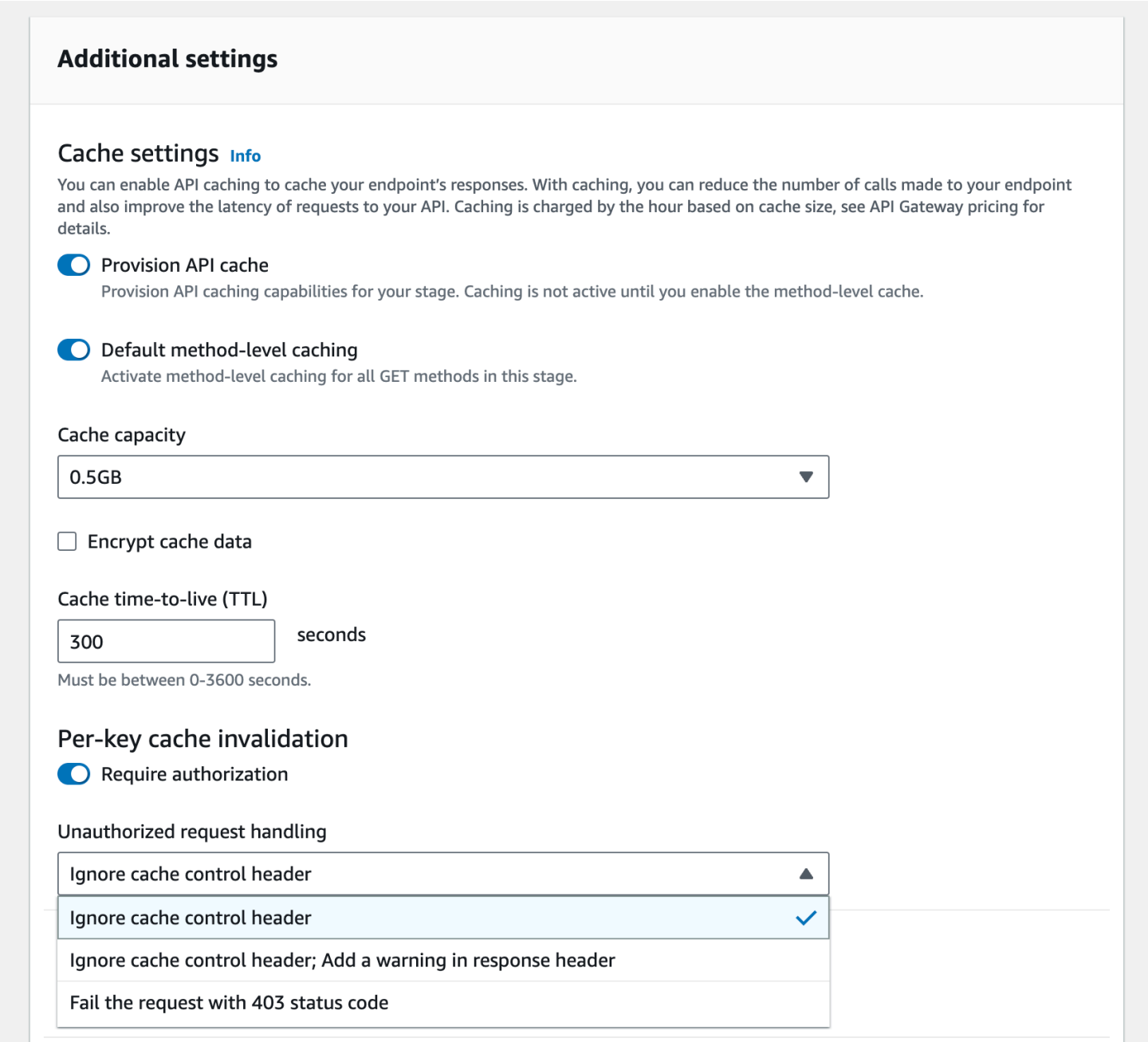
```
        ]
      }
    ]
  }
```

Questa politica consente al servizio di esecuzione API Gateway di invalidare la cache per le richieste sulla risorsa (o sulle risorse) specificate. Per specificare un gruppo di risorse mirate, utilizzate un carattere jolly (*) per `account-id` e altre voci nel ARN valore di. `api-id` `Resource` Per ulteriori informazioni su come impostare le autorizzazioni per il servizio di esecuzione API Gateway, vedere. Controlla l'accesso a un REST API con IAM autorizzazioni

Se non imponi una `InvalidateCache` policy (o scegli la casella di controllo Richiedi autorizzazione nella console), qualsiasi client può invalidare la cache. API Se la maggior parte o tutti i client invalidano la API cache, ciò potrebbe aumentare significativamente la latenza della cache. API

Quando la policy è attiva, il caching è abilitato e l'autorizzazione è richiesta.

È possibile controllare il modo in cui vengono gestite le richieste non autorizzate scegliendo un'opzione da Gestione delle richieste non autorizzate nella console Gateway. API



Le tre opzioni risultano nei comportamenti seguenti:

- Fail the request with 403 status code (Richiesta non riuscita con codice stato 403): restituisce una risposta di tipo non autorizzato 403.

  Per impostare questa opzione utilizzando, usa. API `FAIL_WITH_403`
- Ignore cache control header; Add a warning in response header (Ignora intestazione di controllo cache; aggiungi avviso in intestazione risposta): elabora la richiesta e aggiunge un'intestazione di avviso nella risposta.

Per impostare questa opzione usandoAPI, usa`SUCCEED_WITH_RESPONSE_HEADER`.

- Ignore cache control header (Ignora intestazione di controllo cache): elabora la richiesta senza aggiungere un'intestazione di avviso nella risposta.

  Per impostare questa opzione usandoAPI, usa`SUCCEED_WITHOUT_RESPONSE_HEADER`.

Compressione del payload per REST APIs in API Gateway

APIGateway consente al cliente di chiamarvi API con payload compressi utilizzando una delle codifiche di contenuto supportate. Per impostazione predefinita, API Gateway supporta la decompressione del payload di richiesta del metodo. Tuttavia, è necessario configurare in modo API da abilitare la compressione del payload del metodo di risposta.

Per abilitare la compressione su un `API`, impostate la `minimumCompressionsSize`proprietà su un numero intero non negativo compreso tra 0 e 10485760 (10 milioni di byte) quando create o dopo aver creato il. API API Per disabilitare la compressione suAPI, impostatela su null o rimuovetela del tutto`minimumCompressionSize`. È possibile abilitare o disabilitare la compressione per un API utilizzando la console API Gateway, il o il AWS CLI Gateway. API REST API

Se desideri che la compressione venga applicata a payload di qualsiasi dimensione, imposta il valore di `minimumCompressionSize` su zero. La compressione di dati di piccole dimensioni può tuttavia comportare un aumento della dimensione finale dei dati. Inoltre, la compressione in API Gateway e la decompressione nel client potrebbero aumentare la latenza complessiva e richiedere più tempi di elaborazione. È necessario eseguire dei test case con il proprio API per determinare un valore ottimale.

Il client può inviare una API richiesta con un payload compresso e un'`Content-Encoding`intestazione appropriata per consentire a API Gateway di decomprimere e applicare i modelli di mappatura applicabili, prima di passare la richiesta all'endpoint di integrazione. Dopo aver abilitato la compressione e implementata, API il client può ricevere una API risposta con un payload compresso se specifica un'intestazione appropriata nella richiesta del metodo. `Accept-Encoding`

Quando l'endpoint di integrazione prevede e restituisce payload non compressi, qualsiasi modello di mappatura configurato per un JSON payload non compresso è applicabile al payload compresso. JSON Per un payload di richiesta con metodo compresso, API Gateway decomprime il payload, applica il modello di mappatura e passa la richiesta mappata all'endpoint di integrazione. Per un payload di risposta di integrazione non compresso, API Gateway applica il modello di mappatura, comprime il payload mappato e restituisce il payload compresso al client.

Argomenti

- Abilita la compressione del API payload API per un in Gateway
- Chiama un API metodo con un payload compresso in Gateway API
- Ricevi una API risposta con un payload compresso in Gateway API

Abilita la compressione del API payload API per un in Gateway

È possibile abilitare la compressione per un API utilizzando la console API Gateway, il AWS CLI, o un AWS SDK.

Per una versione esistenteAPI, è necessario implementare la compressione API dopo aver abilitato la compressione affinché la modifica abbia effetto. Per una nuova versioneAPI, è possibile distribuirla API dopo aver completato la API configurazione.

> **Note**
>
> La codifica del contenuto con la massima priorità deve essere quella supportata da Gateway. API In caso contrario, la compressione non viene applicata al payload della risposta.

Argomenti

- Abilita la compressione del payload per un utilizzo della console Gateway API API
- Abilita la compressione del payload per un API utilizzo di AWS CLI
- Codifiche dei contenuti supportate da API Gateway

Abilita la compressione del payload per un utilizzo della console Gateway API API

La procedura seguente descrive come abilitare la compressione del payload per unAPI.

Per abilitare la compressione del payload utilizzando la API console Gateway

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegline uno esistente API o creane uno nuovo.
3. Nel riquadro di navigazione principale, scegli APIimpostazioni.
4. Nella sezione dei APIdettagli, scegli Modifica.

5. Attiva l'opzione Codifica contenuto per abilitare la compressione del payload. In Dimensione corpo minima, immetti un numero per la dimensione di compressione minima (in byte). Per disattivare la compressione, disattiva l'opzione Codifica contenuto.
6. Seleziona Salvataggio delle modifiche.

Abilita la compressione del payload per un API utilizzo di AWS CLI

Per utilizzare il AWS CLI per creare una nuova compressione API e abilitare la compressione, chiama il `create-rest-api`comando come segue:

```
aws apigateway create-rest-api \
    --name "My test API" \
    --minimum-compression-size 0
```

Per utilizzare l'opzione AWS CLI per abilitare la compressione su un sistema esistenteAPI, chiamate il `update-rest-api`comando come segue:

```
aws apigateway update-rest-api \
    --rest-api-id 1234567890 \
    --patch-operations op=replace,path=/minimumCompressionSize,value=0
```

La proprietà `minimumCompressionSize` ha un valore intero non negativo compreso tra 0 e 10485760 (10M byte). Misura la soglia di compressione. Se la dimensione del payload è inferiore a questo valore, la compressione o la decompressione non vengono applicate al payload. Impostando il valore su zero, la compressione viene applicata per qualsiasi dimensione di payload.

Per utilizzare il comando AWS CLI per disabilitare la compressione, chiamate il `update-rest-api`comando come segue:

```
aws apigateway update-rest-api \
    --rest-api-id 1234567890 \
    --patch-operations op=replace,path=/minimumCompressionSize,value=
```

È anche possibile impostare `value` su una stringa vuota `""` o omettere completamente la proprietà `value` nella chiamata precedente.

Codifiche dei contenuti supportate da API Gateway

APIGateway supporta le seguenti codifiche di contenuto:

- `deflate`
- `gzip`
- `identity`

APIGateway supporta anche il seguente formato di `Accept-Encoding` intestazione, in base alla specifica RFC7231:

- `Accept-Encoding:deflate,gzip`
- `Accept-Encoding:`
- `Accept-Encoding:*`
- `Accept-Encoding:deflate;q=0.5,gzip;q=1.0`
- `Accept-Encoding:gzip;q=1.0,identity;q=0.5,*;q=0`

Chiama un API metodo con un payload compresso in Gateway API

Per effettuare una API richiesta con un payload compresso, il client deve impostare l'`Content-Encoding`intestazione con una delle codifiche di contenuto supportate.

Supponiamo che tu sia un API cliente e desideri chiamare il metodo (). PetStore API `POST /pets` Non chiamate il metodo utilizzando il seguente JSON output:

```
POST /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Content-Length: ...

{
  "type": "dog",
  "price": 249.99
}
```

Puoi invece chiamare il metodo con lo stesso payload compresso utilizzando la GZIP codifica:

```
POST /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Content-Encoding:gzip
Content-Length: ...

???RPP*?,HU?RPJ?OW??e&???L,?,-y?j
```

Quando API Gateway riceve la richiesta, verifica se la codifica del contenuto specificata è supportata. Tenta quindi di decomprimere il payload con la codifica di contenuto specificata. Se la decompressione ha esito positivo, la richiesta viene inviata all'endpoint di integrazione. Se la codifica specificata non è supportata o il payload fornito non è compresso con la codifica specificata, API Gateway restituisce la risposta di errore. `415 Unsupported Media Type` L'errore non viene registrato in CloudWatch Logs, se si verifica nella fase iniziale della decompressione prima che la fase e l'altra vengano identificate. API

Ricevi una API risposta con un payload compresso in Gateway API

Quando si effettua una richiesta su un dispositivo abilitato alla compressioneAPI, il client può scegliere di ricevere un payload di risposta compressa in un formato specifico specificando un'intestazione con una `Accept-Encoding` codifica del contenuto supportata.

APIGateway comprime il payload di risposta solo quando sono soddisfatte le seguenti condizioni:

- La richiesta in ingresso ha l'intestazione `Accept-Encoding` con una codifica di contenuto e un formato supportati.

> **Note**
>
> Se l'intestazione non è impostata, il valore predefinito è quello `*` definito in 7231. RFC In tal caso, API Gateway non comprime il payload. Alcuni browser o client possono aggiungere `Accept-Encoding` (ad esempio, `Accept-Encoding:gzip, deflate, br`) automaticamente alle richieste abilitate per la compressione. Ciò può attivare la compressione del payload in Gateway. API Senza una specifica esplicita dei valori di `Accept-Encoding` intestazione supportati, API Gateway non comprime il payload.

- `minimumCompressionSize`È impostato su per abilitare la compressione. API
- La risposta di integrazione non ha un'intestazione `Content-Encoding`.
- La dimensione di un payload di risposta di integrazione, dopo l'applicazione del modello di mappatura applicabile, è maggiore o uguale al valore di `minimumCompressionSize` specificato.

APIGateway applica qualsiasi modello di mappatura configurato per la risposta di integrazione prima di comprimere il payload. Se la risposta di integrazione contiene un'`Content-Encoding`intestazione, API Gateway presume che il payload della risposta di integrazione sia già compresso e salta l'elaborazione della compressione.

Un esempio è l' PetStore APIesempio e la seguente richiesta:

```
GET /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Accept: application/json
```

Il backend risponde alla richiesta con un JSON payload non compresso simile al seguente:

```
200 OK

[
  {
    "id": 1,
    "type": "dog",
    "price": 249.99
  },
  {
    "id": 2,
    "type": "cat",
    "price": 124.99
  },
  {
    "id": 3,
    "type": "fish",
    "price": 0.99
  }
]
```

Per ricevere questo output come payload compresso, il API client può inviare una richiesta come segue:

```
GET /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Accept-Encoding:gzip
```

Il client riceve la risposta con un'`Content-Encoding`intestazione e un payload con GZIP codifica simili ai seguenti:

```
200 OK
Content-Encoding:gzip
...
```

```
◆◆◆RP◆

J◆)JV
◆:P^IeA*◆◆◆◆◆◆+(◆L ◆X◆YZ◆ku0L0B7!9◆◆C#◆&◆◆◆◆Y◆◆a◆◆◆^◆X
```

Quando il payload di risposta è compresso, solo la dimensione dei dati compressi viene fatturata per il trasferimento dati.

Distribuisci le tue API REST ai clienti in API Gateway

In questa sezione vengono fornite informazioni dettagliate sulla distribuzione delle API Gateway ai clienti. La distribuzione dell'API include la generazione di SDK per i clienti per il download e l'integrazione con le relative applicazioni client, la documentazione dell'API in modo che i clienti sappiano come chiamarla dalle loro applicazioni client, rendere l'API disponibile come parte delle offerte di prodotti.

Argomenti

- Piani di utilizzo e API chiavi per REST APIs in API Gateway
- Documentazione per REST APIs in API Gateway
- Genera SDKs per REST APIs in API Gateway
- Vendi le tue API API Gateway tramite Marketplace AWS

Piani di utilizzo e API chiavi per REST APIs in API Gateway

Dopo aver creato, testato e distribuito i tuoiAPIs, puoi utilizzare i piani di utilizzo di API Gateway per renderli disponibili come offerte di prodotti per i tuoi clienti. Puoi configurare piani e API chiavi di utilizzo per consentire ai clienti di accedere a determinati APIs piani di utilizzo e iniziare a limitare le richieste a quelle APIs basate su limiti e quote definiti. Queste possono essere impostate a livello di o API API metodo.

Cosa sono i piani e API le chiavi di utilizzo?

Un piano di utilizzo specifica chi può accedere a una o più API fasi e metodi implementati e, facoltativamente, imposta la frequenza di richieste target per iniziare a limitare le richieste. Il piano utilizza API le chiavi per identificare API i clienti e chi può accedere alle fasi associate a ciascuna chiave. API

APIle chiavi sono valori di stringhe alfanumeriche che distribuisci ai clienti sviluppatori di applicazioni per concedere l'accesso ai tuoi. API Puoi utilizzare API le chiavi insieme agli autorizzatori Lambda, ai IAMruoli o ad Amazon Cognito per controllare l'accesso al tuo. APIs APIGateway può generare API chiavi per tuo conto oppure puoi importarle da un file. CSV È possibile generare una API chiave in API Gateway o importarla in API Gateway da una fonte esterna. Per ulteriori informazioni, consulta the section called “Configura API le chiavi utilizzando la console Gateway API”.

Una API chiave ha un nome e un valore. (I termini «APIchiave» e "valore API chiave» sono spesso usati in modo intercambiabile.) Il nome non può superare 1024 caratteri. Il valore è una stringa alfanumerica tra 20 e 128 caratteri, ad esempio, `apikey1234abcdefghij0123456789`.

> **Important**
>
> APIi valori chiave devono essere unici. Se si tenta di creare due API chiavi con nomi diversi e lo stesso valore, API Gateway le considera come la stessa API chiave.
> Una API chiave può essere associata a più di un piano di utilizzo. Un piano di utilizzo può essere associato a più fasi. Tuttavia, una determinata API chiave può essere associata a un solo piano di utilizzo per ogni fase del tuoAPI.

Un limite di throttling imposta il punto di destinazione in cui dovrebbe iniziare la limitazione delle richieste. Questo può essere impostato a livello di API metodo API o.

Un limite di quota imposta il numero massimo previsto di richieste con una determinata API chiave che possono essere inviate entro un intervallo di tempo specificato. È possibile configurare API metodi individuali per richiedere l'autorizzazione delle API chiavi in base alla configurazione del piano di utilizzo.

I limiti di limitazione e di quota si applicano alle richieste di API chiavi individuali che vengono aggregate in tutte le API fasi di un piano di utilizzo.

> **Note**
>
> I piani di utilizzo della limitazione (della larghezza di banda della rete) e le quote non sono limiti rigidi e vengono applicati sulla base del miglior tentativo. In alcuni casi, i client possono superare le quote impostate. Non fare affidamento sulle quote del piano di utilizzo o sulla limitazione per controllare i costi o bloccare l'accesso a un. API Prendi in considerazione l'utilizzo Budget AWSper monitorare i costi e AWS WAFgestire le richieste. API

Le migliori pratiche per API chiavi e piani di utilizzo

Di seguito sono riportate le best practice suggerite da seguire per l'utilizzo di API chiavi e piani di utilizzo.

> **Important**
>
> - Non utilizzare API chiavi per l'autenticazione o l'autorizzazione per controllare l'accesso al tuoAPIs. Se ne hai più di uno APIs in un piano di utilizzo, un utente con una API chiave valida per una di esse API in quel piano di utilizzo può accedere APIs a tutte le chiavi incluse in quel piano di utilizzo. Invece, per controllare l'accesso al tuoAPI, usa un IAM ruolo, un autorizzatore Lambda o un pool di utenti Amazon Cognito.
> - Usa API le chiavi generate da GatewayAPI. APIle chiavi non devono includere informazioni riservate; i client in genere le trasmettono in intestazioni che possono essere registrate.

- Se utilizzi un portale per sviluppatori per pubblicare i tuoiAPIs, tieni presente che tutti i tuoi contenuti APIs inclusi in un determinato piano di utilizzo possono essere sottoscritti dai clienti, anche se non li hai resi visibili ai tuoi clienti.
- In alcuni casi, i client possono superare le quote impostate. Non fare affidamento sui piani di utilizzo per controllare i costi. Valuta la possibilità Budget AWSdi utilizzarlo per monitorare i costi e AWS WAFgestire le API richieste.
- Dopo aver aggiunto una API chiave a un piano di utilizzo, il completamento dell'operazione di aggiornamento potrebbe richiedere alcuni minuti.

Passaggi per configurare un piano di utilizzo in API Gateway

I passaggi seguenti descrivono come, in qualità di API proprietario, crei e configuri un piano di utilizzo per i tuoi clienti.

Per configurare un piano di utilizzo

1. Creane uno o piùAPIs, configura i metodi per richiedere una API chiave e distribuisci APIs le due fasi.

   Per ulteriori informazioni su come configurare un metodo per richiedere una API chiave, vederethe section called “Richiede una API chiave per un metodo”.

2. Genera o importa API chiavi da distribuire agli sviluppatori di applicazioni (i tuoi clienti) che utilizzeranno le tueAPI.

   Per ulteriori informazioni su come generare o importare API chiavi, consultathe section called "Crea una API chiave".

3. Crea il piano di utilizzo con i limiti di limitazione e quota desiderati, quindi associa API fasi e API chiavi al piano di utilizzo.

   Per ulteriori informazioni su come creare il piano di utilizzo, consulta. the section called "Creazione di un piano di utilizzo"

I chiamanti di API devono fornire una API chiave assegnata nell'`x-api-key`intestazione nelle richieste a. API

Per un AWS CloudFormation modello completo che crea e associa un piano di utilizzo a una nuova fase e una API chiave, consulta. the section called "Crea e configura API chiavi e piani di utilizzo con AWS CloudFormation"

> **Note**
>
> Per includere API i metodi in un piano di utilizzo, è necessario configurare API i singoli metodi in modo che richiedano una API chiave. Per informazioni sulle best practice da prendere in considerazione, consulta the section called "Le migliori pratiche per API chiavi e piani di utilizzo".

Scegli una fonte API chiave in API Gateway

Quando associ un piano di utilizzo a un metodo API e abiliti API le chiavi sui API metodi, ogni richiesta in arrivo API deve contenere una APIchiave. APIGateway legge la chiave e la confronta con le chiavi del piano di utilizzo. In caso di corrispondenza, API Gateway limita le richieste in base al limite e alla quota di richieste del piano. In caso contrario, genera un'eccezione `InvalidKeyParameter` e l'intermediario riceve una risposta `403 Forbidden`.

Il tuo API Gateway API può ricevere API le chiavi da una delle due fonti seguenti:

HEADER

Distribuisci API le chiavi ai tuoi clienti e richiedi loro di passare la API chiave come `X-API-Key` intestazione di ogni richiesta in arrivo.

AUTHORIZER

Hai un autorizzatore Lambda che restituisce la API chiave come parte della risposta di autorizzazione. Per ulteriori informazioni sulla risposta delle autorizzazioni, consulta the section called "Output da un autorizzatore API Gateway Lambda".

> **Note**
>
> Per informazioni sulle best practice da prendere in considerazione, consulta the section called "Le migliori pratiche per API chiavi e piani di utilizzo".

Per scegliere una fonte di API chiavi per un utente API utilizzando la API console Gateway

1. Accedere alla console API Gateway.
2. Scegline uno esistente API o creane uno nuovo.
3. Nel riquadro di navigazione principale, scegli APIimpostazioni.
4. Nella sezione dei APIdettagli, scegli Modifica.
5. In Fonte API chiave, seleziona `Header` o `Authorizer` dall'elenco a discesa.
6. Seleziona Salvataggio delle modifiche.

Per scegliere una fonte API chiave per un utilizzo API di AWS CLI, chiamate il `update-rest-api`comando come segue:

```
aws apigateway update-rest-api --rest-api-id 1234123412 --patch-operations
 op=replace,path=/apiKeySource,value=AUTHORIZER
```

Per fare in modo che il client invii una API chiave, impostate `value` to `HEADER` nel CLI comando precedente.

Per scegliere una fonte di API chiavi per un utente API utilizzando il API Gateway RESTAPI, chiamate `restapi:update`come segue:

```
PATCH /restapis/fugvjdxtri/ HTTP/1.1
Content-Type: application/json
Host: apigateway.us-east-1.amazonaws.com
X-Amz-Date: 20160603T205348Z
Authorization: AWS4-HMAC-SHA256 Credential={access_key_ID}/20160603/us-east-1/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature={sig4_hash}

{
  "patchOperations" : [
    {
        "op" : "replace",
        "path" : "/apiKeySource",
        "value" : "HEADER"
    }
  ]
}
```

Per fare in modo che un autorizzatore restituisca una API chiave, `value` impostala su `AUTHORIZER` nell'`patchOperations`input precedente.

A seconda del tipo di sorgente di API chiave scelto, utilizzate una delle seguenti procedure per utilizzare API chiavi provenienti dall'intestazione o chiavi restituite dall'APIautorizzatore nella chiamata del metodo:

Per utilizzare chiavi fornite dall'intestazione: API

1. Createne uno API con API i metodi desiderati, quindi distribuitelo su uno stage. API
2. Crea un nuovo piano di utilizzo o scegline uno esistente. Aggiungi la API fase di distribuzione al piano di utilizzo. Allega una API chiave al piano di utilizzo o scegli una API chiave esistente nel piano. Annota il valore API chiave scelto.
3. Imposta API i metodi per richiedere una API chiave.
4. Ridistribuiscili nella API stessa fase. Se lo distribuisci in una nuova fase, assicurati di aggiornare il piano di utilizzo per collegare la nuova fase. API API

Il client può ora chiamare i API metodi fornendo all'`x-api-key`intestazione la API chiave scelta come valore dell'intestazione.

Per utilizzare chiavi fornite dall'autorizzatore: API

1. Creane uno API con API i metodi desiderati, quindi distribuiscilo su una fase. API
2. Crea un nuovo piano di utilizzo o scegline uno esistente. Aggiungi la API fase di distribuzione al piano di utilizzo. Allega una API chiave al piano di utilizzo o scegli una API chiave esistente nel piano. Annota il valore API chiave scelto.
3. Crea una funzione di autorizzazione Lambda basata su token. Includi `usageIdentifierKey:{api-key}` come proprietà a livello di root della risposta di autorizzazione. Per istruzioni sulla creazione di un autorizzatore basato su token, consulta. the section called “Esempio di funzione Lambda `TOKEN` dell'autorizzatore”
4. Imposta API i metodi per richiedere una API chiave e abilita anche l'autorizzatore Lambda sui metodi.
5. Ridistribuiscili nella stessa faseAPI. Se lo distribuisci in una nuova fase, assicurati di aggiornare il piano di utilizzo per collegare la nuova fase. API API

Il client può ora chiamare i metodi API richiesti dalla chiave senza fornire esplicitamente alcuna chiave. API La chiave restituita dall'autorizzatore API viene utilizzata automaticamente.

Configura API le chiavi utilizzando la console Gateway API

Per configurare API le chiavi, procedi come segue:

- Configura API i metodi per richiedere una API chiave.
- Crea o importa una API chiave per loro API in una regione.

Prima di impostare API le chiavi, è necessario averne creata una API e distribuita in una fase. Dopo aver creato un valore API chiave, non può essere modificato.

Per istruzioni su come creare e distribuire e utilizzare la console API Gateway, vedete rispettivamente Sviluppa REST APIs in API Gateway eImplementazione REST APIs in Gateway API. API

Dopo aver creato una API chiave, è necessario associarla a un piano di utilizzo. Per ulteriori informazioni, consulta Crea, configura e testa i piani di utilizzo con la console API Gateway.

> **Note**
>
> Per informazioni sulle best practice da prendere in considerazione, consulta the section called “Le migliori pratiche per API chiavi e piani di utilizzo”.

Argomenti

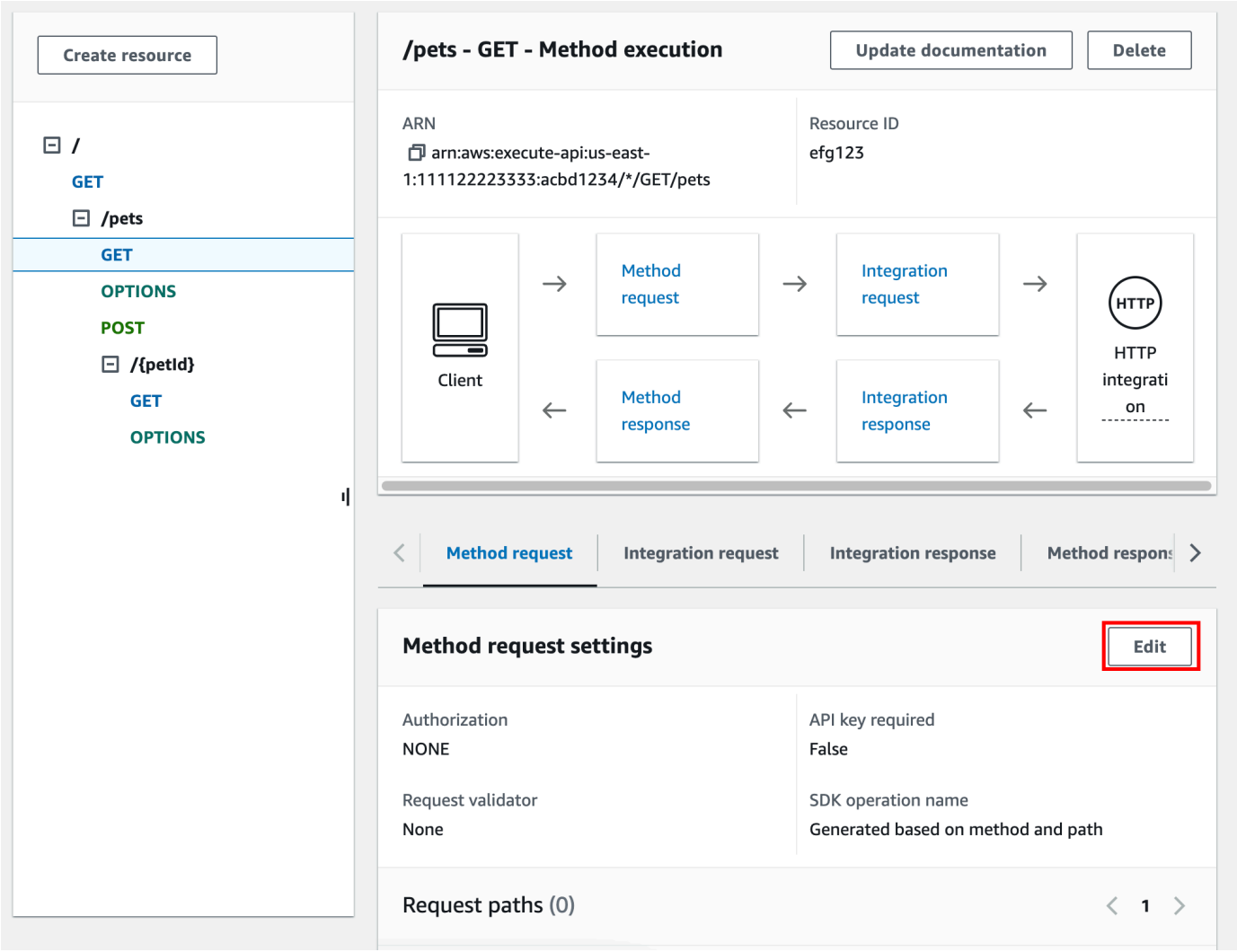
- Richiede una API chiave per un metodo
- Crea una API chiave
- Importa API chiavi

Richiede una API chiave per un metodo

La procedura seguente descrive come configurare un API metodo per richiedere una API chiave.

Per configurare un API metodo in modo da richiedere una API chiave

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli un RESTAPI.
3. Nel pannello di navigazione principale del API Gateway, scegli Risorse.
4. In Resources (Risorse) creare un nuovo metodo o sceglierne uno esistente.
5. Nella scheda Richiesta metodo, in Impostazioni richiesta metodo, scegli Modifica.



6. Seleziona APIla chiave richiesta.
7. Selezionare Salva.
8. Implementa o ridistribuisci il API affinché il requisito abbia effetto.

Se l'opzione APIchiave richiesta è impostata su `false` e non si eseguono i passaggi precedenti, per il metodo non viene utilizzata alcuna API chiave associata a una API fase.

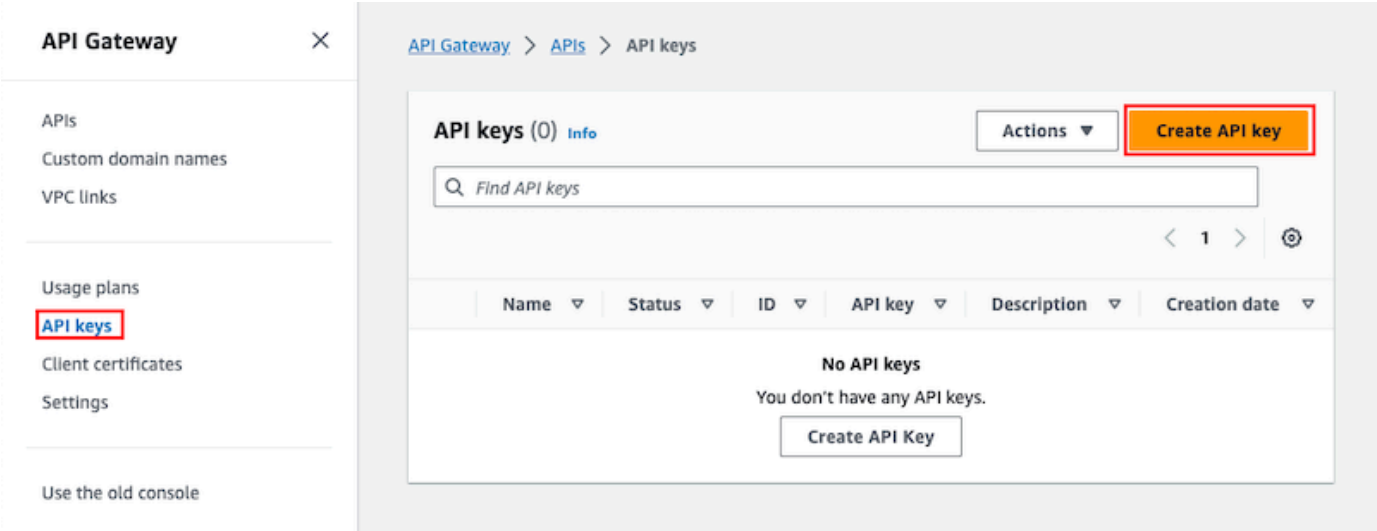
Crea una API chiave

Se hai già creato o importato API chiavi da utilizzare con i piani di utilizzo, puoi saltare questa procedura e la successiva.

Per creare una chiave API

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.

2. Scegli un RESTAPI.
3. Nel pannello di navigazione principale del API Gateway, scegli i APItasti.
4. Scegli Crea API chiave.



5. In Nome, immetti un nome.
6. (Facoltativo) In Description (Descrizione), immettere una descrizione.
7. Per APIchiave, scegli Generazione automatica per fare in modo che API Gateway generi il valore della chiave oppure scegli Personalizzato per creare il tuo valore di chiave.
8. Selezionare Salva.

Importa API chiavi

La procedura seguente descrive come importare API le chiavi da utilizzare con i piani di utilizzo.

Per importare API le chiavi

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli un RESTAPI.
3. Nel riquadro di navigazione principale, scegli i APItasti.
4. Scegli il menu a discesa Azioni, quindi scegli Importa API chiavi.
5. Per caricare un file di chiavi con valori separati da virgole, scegli Scegli file. Puoi anche immettere le chiavi nell'editor di testo. Per informazioni sul formato del file, consulta the section called “APIFormato del file API della chiave Gateway”.
6. Scegli Errore su avvertenze per arrestare l'importazione se si verifica un errore oppure Ignora avvisi per continuare a importare voci di chiavi valide in caso di presenza di avvisi.

7. Scegli Importa per importare le tue API chiavi.

Crea, configura e testa i piani di utilizzo con la console API Gateway

Prima di creare un piano di utilizzo, assicurati di aver impostato le API chiavi desiderate. Per ulteriori informazioni, consulta Configura API le chiavi utilizzando la console Gateway API.

Questa sezione descrive come creare e utilizzare un piano di utilizzo utilizzando la console API Gateway.

Argomenti

- Esegui la migrazione API ai piani di utilizzo predefiniti (se necessario)
- Creazione di un piano di utilizzo
- Test di un piano di utilizzo
- Gestione di un piano di utilizzo

Esegui la migrazione API ai piani di utilizzo predefiniti (se necessario)

Se hai iniziato a utilizzare API Gateway dopo il lancio della funzionalità dei piani di utilizzo l'11 agosto 2016, avrai automaticamente i piani di utilizzo abilitati in tutte le regioni supportate.

Se hai iniziato a utilizzare API Gateway prima di tale data, potresti dover migrare ai piani di utilizzo predefiniti. Ti verrà proposta l'opzione Enable Usage Plans (Abilita piani di utilizzo) prima di usare i piani di utilizzo per la prima volta nella regione selezionata. Quando abiliti questa opzione, hai piani di utilizzo predefiniti creati per ogni API fase unica associata alle API chiavi esistenti. Nel piano di utilizzo predefinito, inizialmente non vengono impostati limiti di limitazione o di quota e le associazioni tra le API chiavi e le API fasi vengono copiate nei piani di utilizzo. APISi comporta come prima. Tuttavia, è necessario utilizzare la `UsagePlan``apiStages`proprietà per associare i valori di API fase specificati (`apiIdandstage`) alle API chiavi incluse (via `UsagePlanKey`), anziché utilizzare la ApiKey`stageKeys`proprietà.

Per verificare se hai già effettuato la migrazione ai piani di utilizzo predefiniti, usa il `get-account`CLIcomando. Nell'output del comando, l'elenco `features` include una voce `"UsagePlans"` quando i piani di utilizzo sono abilitati.

Puoi anche migrare i tuoi piani di utilizzo APIs ai piani di utilizzo predefiniti utilizzando AWS CLI quanto segue:

Per migrare ai piani di utilizzo predefiniti utilizzando il AWS CLI

1. Chiama questo CLI comando: `update-account`.
2. Per il `cli-input-json` parametro, usa quanto segueJSON:

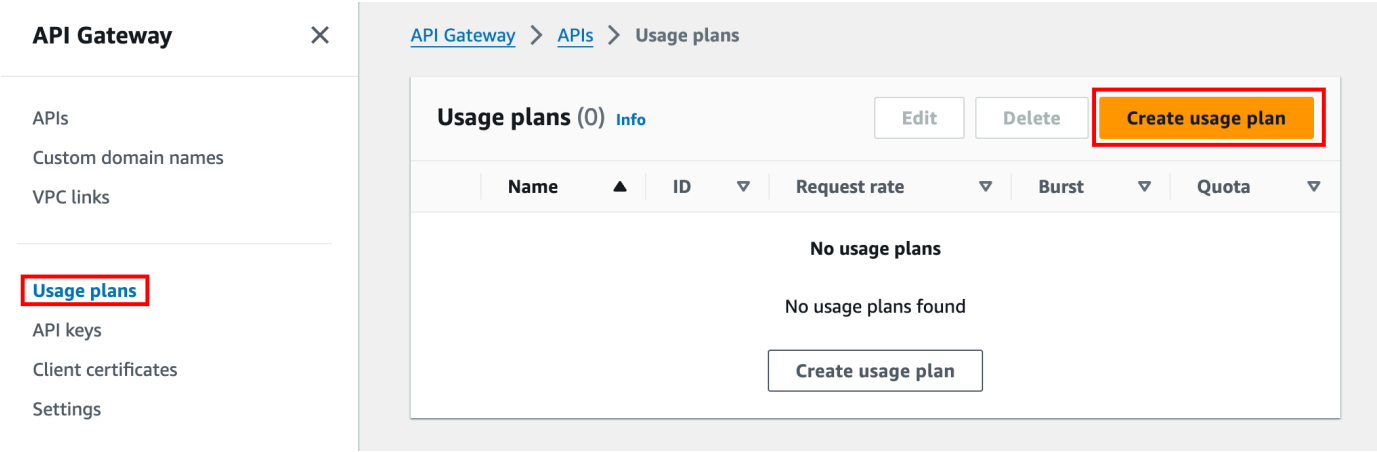
```
[
    {
        "op": "add",
        "path": "/features",
        "value": "UsagePlans"
    }
]
```

Creazione di un piano di utilizzo

La procedura seguente illustra come creare un piano di utilizzo.

Per creare un piano di utilizzo

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel pannello di navigazione principale del API Gateway, scegli Piani di utilizzo, quindi scegli Crea piano di utilizzo.

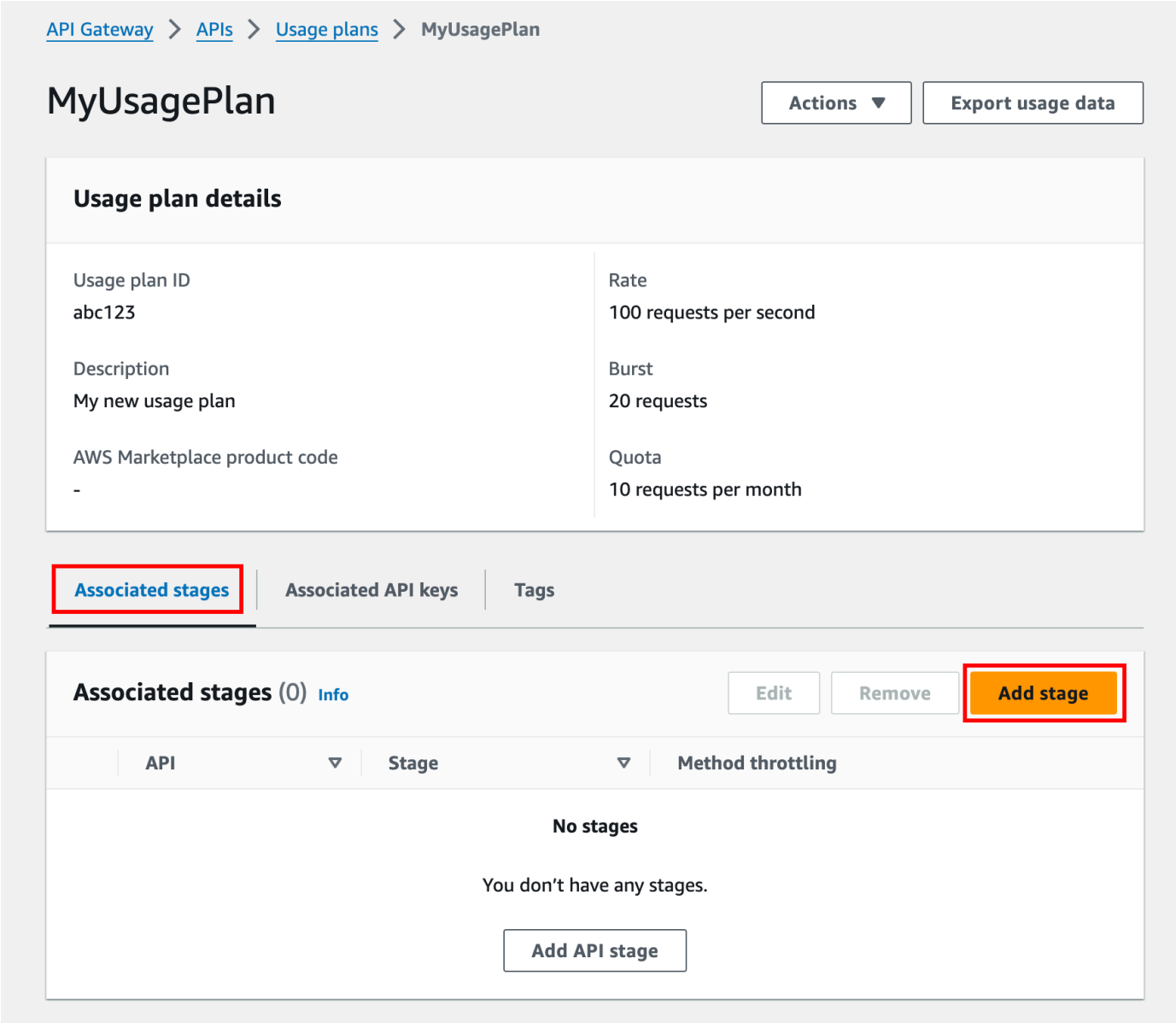


3. In Nome, immetti un nome.
4. (Facoltativo) In Description (Descrizione), immettere una descrizione.
5. Per impostazione predefinita, i piani di utilizzo abilitano la limitazione (della larghezza di banda della rete). Immetti un valore in Tariffa e Ottimizzazione per il tuo piano di utilizzo. Scegli Throttling per disattivare la limitazione (della larghezza di banda della rete).

6. Per impostazione predefinita, i piani di utilizzo abilitano una quota per un periodo di tempo. In Richieste, immetti il numero totale di richieste che un utente può effettuare nel periodo di validità del tuo piano di utilizzo. Scegli Quota per disattivare la quota.
7. Scegli Crea piano di utilizzo.

Aggiunta di una fase al piano di utilizzo

1. Seleziona il piano di utilizzo.
2. Nella scheda Fasi associate, scegli Aggiungi fase.

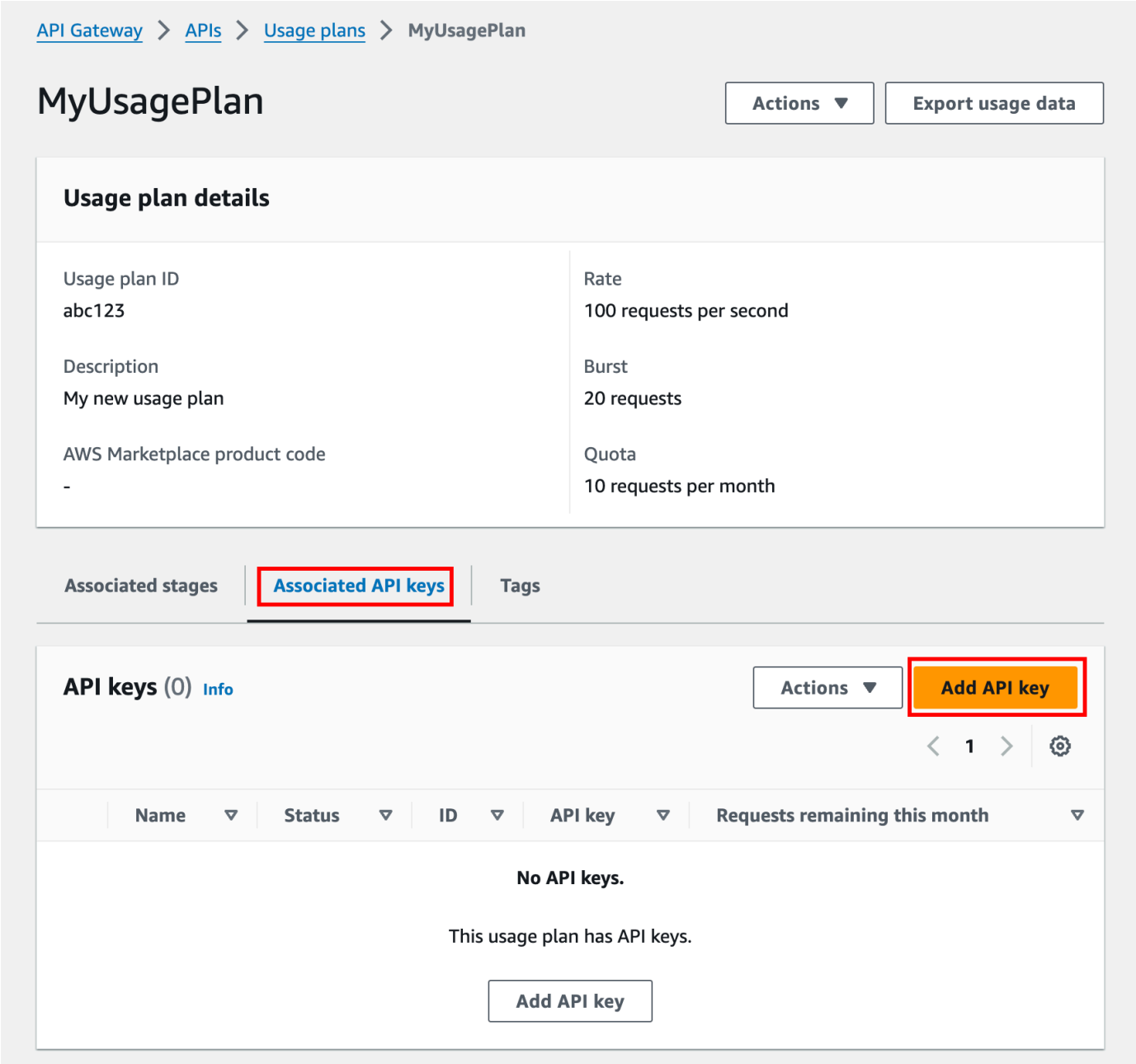


3. Per API, seleziona unAPI.

4. In Fase, seleziona una fase.
5. (Facoltativo) Per attivare la limitazione (della larghezza di banda della rete) a livello di metodo, esegui le operazioni indicate di seguito:
   a. Scegli Throttling a livello di metodo e quindi Aggiungi metodo.
   b. Per Risorsa, seleziona una risorsa dalla tuaAPI.
   c. Per Metodo, seleziona un metodo dal tuoAPI.
   d. Immetti un valore in Tariffa e Ottimizzazione per il tuo piano di utilizzo.
6. Scegli Aggiungi al piano di utilizzo.

Aggiunta di una chiave al piano di utilizzo

1. Nella scheda APIChiavi associate, scegli Aggiungi API chiave.



2. a. Per associare una chiave esistente al tuo piano di utilizzo, seleziona Aggiungi chiave esistente, quindi seleziona la chiave esistente nel menu a discesa.

   b. Per creare una nuova API chiave, seleziona Crea e aggiungi nuova chiave, quindi crea una nuova chiave. Per ulteriori informazioni su come creare una nuova chiave, consulta Crea una API chiave.

3. Scegli Aggiungi API chiave.

Test di un piano di utilizzo

Per testare il piano di utilizzo, puoi utilizzare un AWS SDK o un REST API client come Postman. AWS CLI Per un esempio di uso di Postman per testare il piano di utilizzo, consultare Test dei piani di utilizzo.

Gestione di un piano di utilizzo

La gestione di un piano di utilizzo implica il monitoraggio delle quote utilizzate e rimanenti in un determinato periodo di tempo, se necessario, e l'estensione delle quote rimanenti di una quantità specificata. Le procedure seguenti descrivono come monitorare le quote.

Per monitorare le quote utilizzate e rimanenti

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel pannello di navigazione principale del API Gateway, scegli Piani di utilizzo.
3. Seleziona un piano di utilizzo.
4. Scegli la scheda APIChiavi associate per visualizzare il numero di richieste rimanenti per il periodo di tempo per ciascuna chiave.
5. (Facoltativo) Scegli Esporta dati di utilizzo, quindi scegli una data in Da e una data in A. Quindi scegli JSONo CSVper il formato dei dati esportati, quindi scegli Esporta.

   L'esempio di seguito mostra un file esportato.

```
{
  "thisPeriod": {
    "px1KW6...qBazOJH": [
      [
        0,
        5000
      ],
      [
        0,
        5000
      ],
      [
        0,
        10
      ]
    ]
  },
```

```
  "startDate": "2016-08-01",
  "endDate": "2016-08-03"
}
```

I dati di utilizzo nell'esempio mostrano i dati di utilizzo giornaliero per un API cliente, identificati dalla API chiave (`px1KW6...qBazOJH`), tra il 1° agosto 2016 e il 3 agosto 2016. I dati di utilizzo giornalieri mostrano le quote utilizzate e rimanenti. In questo esempio, l'abbonato non ha ancora utilizzato alcuna quota assegnata e il API proprietario o l'amministratore ha ridotto la quota rimanente da 5000 a 10 il terzo giorno.

Le procedure seguenti descrivono come modificare le quote.

Per estendere le quote rimanenti

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel pannello di navigazione principale del API Gateway, scegli Piani di utilizzo.
3. Seleziona un piano di utilizzo.
4. Scegli la scheda APIChiavi associate per visualizzare il numero di richieste rimanenti per il periodo di tempo per ciascuna chiave.
5. Seleziona una API chiave, quindi scegli Grant usage extension.
6. Immetti un numero di quote in Richieste rimanenti. Puoi aumentare o diminuire le richieste rimanenti per il periodo di validità del tuo piano di utilizzo.
7. Scegli Aggiorna quota.

Configura API le chiavi utilizzando il API Gateway REST API

Per configurare API le chiavi, procedi come segue:

- Configura API i metodi per richiedere una API chiave.
- Crea o importa una API chiave per loro API in una regione.

Prima di impostare API le chiavi, è necessario averne creata una API e distribuita in una fase. Dopo aver creato un valore API chiave, non può essere modificato.

Per le REST API chiamate per creare e distribuire unAPI, vedi `restapi:create`e `deployment:create`, rispettivamente.

> **Note**
>
> Per informazioni sulle best practice da prendere in considerazione, consulta the section called "Le migliori pratiche per API chiavi e piani di utilizzo".

Argomenti

- Richiede una API chiave su un metodo
- Creare o importare API chiavi

Richiede una API chiave su un metodo

Per richiedere una API chiave su un metodo, effettuate una delle seguenti operazioni:

- Invoca `method:put` per creare un metodo. Imposta `apiKeyRequired` un `true` nel payload di richiesta.
- Invoca `method:update` per impostare `apiKeyRequired` su `true`.

Creare o importare API chiavi

Per creare o importare una API chiave, effettuate una delle seguenti operazioni:

- Chiama `apikey:create`per creare una API chiave.
- Chiama `apikey:import`per importare una API chiave da un file. Per il formato file, consulta APIFormato del file API della chiave Gateway.

Non è possibile modificare il valore della nuova API chiave. Per sapere come configurare un piano di utilizzo, consultare Crea, configura e testa i piani di utilizzo utilizzando API Gateway CLI e REST API.

Crea, configura e testa i piani di utilizzo utilizzando API Gateway CLI e REST API

Prima di configurare un piano di utilizzo, è necessario aver già eseguito le seguenti operazioni: impostato i metodi di una persona selezionata API per richiedere API le chiavi, distribuirla o ridistribuirla in una fase e creare o API importare una o più chiavi. API Per ulteriori informazioni, consulta Configura API le chiavi utilizzando il API Gateway REST API.

Per configurare un piano di utilizzo utilizzando il API Gateway RESTAPI, utilizza le seguenti istruzioni, supponendo che tu abbia già creato il piano di utilizzo da aggiungere APIs al piano di utilizzo.

Argomenti

- Migrazione ai piani di utilizzo predefiniti
- Creazione di un piano di utilizzo
- Gestisci un piano di utilizzo utilizzando il AWS CLI
- Test dei piani di utilizzo

Migrazione ai piani di utilizzo predefiniti

Quando si crea un piano di utilizzo per la prima volta, è possibile migrare API le fasi esistenti associate alle API chiavi selezionate verso un piano di utilizzo `account:update`chiamando il seguente organismo:

```
{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/features",
    "value" : "UsagePlans"
  } ]
}
```

Per ulteriori informazioni sulla migrazione delle API fasi associate alle API chiavi, consulta Migrazione ai piani di utilizzo predefiniti nella console Gateway. API

Creazione di un piano di utilizzo

La procedura seguente illustra come creare un piano di utilizzo.

Per creare un piano di utilizzo con REST API

1. Invoca `usageplan:create` per creare un piano di utilizzo. Nel payload, specificate il nome e la descrizione del piano, le API fasi associate, i limiti di tariffa e le quote.

   Prendi nota dell'identificativo del piano di utilizzo. Questo valore servirà nella fase successiva.
2. Esegui una di queste operazioni:

   a. Chiama `usageplankey:create`per aggiungere una API chiave al piano di utilizzo. Specifica `keyId` e `keyType` nel payload.

Per aggiungere altre API chiavi al piano di utilizzo, ripeti la chiamata precedente, una API chiave alla volta.

b. Chiama `apikey:import`per aggiungere una o più API chiavi direttamente al piano di utilizzo specificato. Il payload della richiesta deve contenere i valori API chiave, l'identificatore del piano di utilizzo associato, i flag booleani per indicare che le chiavi sono abilitate per il piano di utilizzo e, possibilmente, i nomi e le descrizioni delle API chiavi.

Il seguente esempio di `apikey:import` richiesta aggiunge tre API chiavi (identificate da `keyname`, `edescription`) a un piano di utilizzo (come identificato da): `usageplanIds`

```
POST /apikeys?mode=import&format=csv&failonwarnings=fase HTTP/1.1
Host: apigateway.us-east-1.amazonaws.com
Content-Type: text/csv
Authorization: ...

key,name, description, enabled, usageplanIds
abcdef1234ghijklmnop8901234567, importedKey_1, firstone,  tRuE, n371pt
abcdef1234ghijklmnop0123456789, importedKey_2, secondone, TRUE, n371pt
abcdef1234ghijklmnop9012345678, importedKey_3,          , true, n371pt
```

Di conseguenza, vengono create tre risorse `UsagePlanKey` che vengono aggiunte a `UsagePlan`.

In questo modo puoi anche aggiungere API chiavi a più di un piano di utilizzo. A questo scopo, modifica ogni valore della colonna `usageplanIds` in una stringa separata da virgole contenente gli identificatori del piano di utilizzo selezionato e racchiusa tra apici (`"n371pt,m282qs"` o `'n371pt,m282qs'`).

> **Note**
>
> Una API chiave può essere associata a più di un piano di utilizzo. Un piano di utilizzo può essere associato a più fasi. Tuttavia, una determinata API chiave può essere associata a un solo piano di utilizzo per ogni fase del tuoAPI.

Gestisci un piano di utilizzo utilizzando il AWS CLI

I seguenti codici di esempio mostrano come aggiungere, rimuovere o modificare le impostazioni di throttling a livello di metodo in un piano di utilizzo invocando il comando `update-usage-plan`.

> **Note**
>
> Assicurati di `us-east-1` impostare il valore della regione appropriato per il tuoAPI.

Per aggiungere o sostituire un limite di tasso per il throttling di una risorsa e un metodo singoli:

```
aws apigateway --region us-east-1 update-usage-plan --usage-plan-id <planId> --patch-
operations
                        op="replace",path="/apiStages/<apiId>:<stage>/
throttle/<resourcePath>/<httpMethod>/rateLimit",value="0.1"
```

Per aggiungere o sostituire un limite di ottimizzazione per il throttling di una risorsa e un metodo singoli:

```
aws apigateway --region us-east-1 update-usage-plan --usage-plan-id <planId>
 --patch-operations op="replace",path="/apiStages/<apiId>:<stage>/
throttle/<resourcePath>/<httpMethod>/burstLimit",value="1"
```

Per rimuovere le impostazioni di throttling a livello di metodo per una risorsa e un metodo singoli:

```
aws apigateway --region us-east-1 update-usage-plan --usage-plan-id <planId>
 --patch-operations op="remove",path="/apiStages/<apiId>:<stage>/
throttle/<resourcePath>/<httpMethod>",value=""
```

Per rimuovere tutte le impostazioni di limitazione a livello di metodo per un: API

```
aws apigateway --region us-east-1 update-usage-plan --usage-plan-id <planId> --patch-
operations op="remove",path="/apiStages/<apiId>:<stage>/throttle ",value=""
```

Ecco un esempio di utilizzo dell'esempio di Pet Store: API

```
aws apigateway --region us-east-1 update-usage-plan --usage-plan-id <planId> --patch-
operations
```

```
                op="replace",path="/apiStages/<apiId>:<stage>/throttle",value='"{\"/
pets/GET\":{\"rateLimit\":1.0,\"burstLimit\":1},\"//GET\":{\"rateLimit\":1.0,
\"burstLimit\":1}}"'
```

Test dei piani di utilizzo

Ad esempio, utilizziamo il PetStore API, che è stato creato inTutorial: creazione di un'API REST mediante l'importazione di un esempio. Supponiamo che API sia configurato per utilizzare una API chiave di`Hiorr45VR...c4GJc`. La procedura seguente illustra come testare un piano di utilizzo.

Per testare il piano di utilizzo

- Effettua una `GET` richiesta sulla risorsa Pets (`/pets`), con i parametri di `?type=...&page=...` query, di API (ad esempio,`xbvxlpijch`) in un piano di utilizzo:

  ```
  GET /testStage/pets?type=dog&page=1 HTTP/1.1
  x-api-key: Hiorr45VR...c4GJc
  Content-Type: application/x-www-form-urlencoded
  Host: xbvxlpijch.execute-api.ap-southeast-1.amazonaws.com
  X-Amz-Date: 20160803T001845Z
  Authorization: AWS4-HMAC-SHA256 Credential={access_key_ID}/20160803/ap-southeast-1/
  execute-api/aws4_request, SignedHeaders=content-type;host;x-amz-date;x-api-key,
   Signature={sigv4_hash}
  ```

  > **Note**
  >
  > È necessario inviare questa richiesta al `execute-api` componente di API Gateway e fornire la API chiave richiesta (ad esempio,`Hiorr45VR...c4GJc`) nell'`x-api-key`intestazione richiesta.

  Se riesce, la risposta restituisce un codice di stato `200 OK` e un payload contenente i risultati richiesti dal back-end: Se dimentichi di impostare l'intestazione `x-api-key` o se la imposti con una chiave sbagliata, ottieni una risposta `403 Forbidden`. Tuttavia, se non hai configurato il metodo per richiedere una API chiave, probabilmente otterrai una `200 OK` risposta indipendentemente dal fatto che tu abbia impostato l'`x-api-key`intestazione correttamente o meno, e i limiti di limitazione e di quota del piano di utilizzo vengono aggirati.

  A volte, quando si verifica un errore interno che impedisce a API Gateway di applicare i limiti o le quote di limitazione del piano di utilizzo per la richiesta, API Gateway soddisfa la richiesta

senza applicare i limiti o le quote di limitazione specificati nel piano di utilizzo. Tuttavia, registra un messaggio di errore. `Usage Plan check failed due to an internal error` CloudWatch Puoi ignorare questi errori occasionali.

Crea e configura API chiavi e piani di utilizzo con AWS CloudFormation

Puoi utilizzare AWS CloudFormation per richiedere API chiavi sui API metodi e creare un piano di utilizzo per unAPI. Il AWS CloudFormation modello di esempio esegue le seguenti operazioni:

- Crea un API gateway API con `POST` metodi `GET` e.
- Richiede una API chiave per i `POST` metodi `GET` and. Questo API riceve le chiavi dall'`X-API-KEY`intestazione di ogni richiesta in arrivo.
- Crea una chiave. API
- Crea un piano di utilizzo per specificare una quota mensile di 1000 richieste al mese, un limite di velocità di limitazione di 100 richieste al secondo e un limite di limitazione di 200 richieste al secondo.
- Specifica un limite di velocità di limitazione a livello di metodo di 50 richieste al secondo e un limite a livello di metodo di 100 richieste al secondo per il metodo `GET`.
- Associa la API fase e la API chiave al piano di utilizzo.

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  StageName:
    Type: String
    Default: v1
    Description: Name of API stage.
  KeyName:
    Type: String
    Default: MyKeyName
    Description: Name of an API key
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Name: keys-api
      ApiKeySourceType: HEADER
  PetsResource:
    Type: 'AWS::ApiGateway::Resource'
```

```
    Properties:
      RestApiId: !Ref Api
      ParentId: !GetAtt Api.RootResourceId
      PathPart: 'pets'
  PetsMethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref PetsResource
      HttpMethod: GET
      ApiKeyRequired: true
      AuthorizationType: NONE
      Integration:
        Type: HTTP_PROXY
        IntegrationHttpMethod: GET
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
  PetsMethodPost:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref PetsResource
      HttpMethod: POST
      ApiKeyRequired: true
      AuthorizationType: NONE
      Integration:
        Type: HTTP_PROXY
        IntegrationHttpMethod: GET
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
  ApiDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn:
      - PetsMethodGet
    Properties:
      RestApiId: !Ref Api
      StageName: !Sub '${StageName}'
  UsagePlan:
    Type: AWS::ApiGateway::UsagePlan
    DependsOn:
      - ApiDeployment
    Properties:
      Description: Example usage plan with a monthly quota of 1000 calls and method-
level throttling for /pets GET
      ApiStages:
        - ApiId: !Ref Api
```

```
          Stage: !Sub '${StageName}'
          Throttle:
            "/pets/GET":
              RateLimit: 50.0
              BurstLimit: 100
      Quota:
        Limit: 1000
        Period: MONTH
      Throttle:
        RateLimit: 100.0
        BurstLimit: 200
      UsagePlanName: "My Usage Plan"
  ApiKey:
    Type: AWS::ApiGateway::ApiKey
    Properties:
      Description: API Key
      Name: !Sub '${KeyName}'
      Enabled: True
  UsagePlanKey:
    Type: AWS::ApiGateway::UsagePlanKey
    Properties:
      KeyId: !Ref ApiKey
      KeyType: API_KEY
      UsagePlanId: !Ref UsagePlan
Outputs:
  ApiRootUrl:
    Description: Root Url of the API
    Value: !Sub 'https://${Api}.execute-api.${AWS::Region}.amazonaws.com/${StageName}'
```

Configura un metodo per utilizzare API le chiavi con una definizione aperta API

È possibile utilizzare una API definizione aperta per richiedere API chiavi su un metodo.

Per ogni metodo, create un oggetto requisito di sicurezza per richiedere una API chiave per richiamare quel metodo. Quindi, definisci `api_key` nella definizione di sicurezza. Dopo aver creato il tuoAPI, aggiungi la nuova API fase al tuo piano di utilizzo.

L'esempio seguente crea un API e richiede una API chiave per i `GET` metodi `POST` and:

OpenAPI 2.0

```
{
  "swagger" : "2.0",
```

```
  "info" : {
    "version" : "2024-03-14T20:20:12Z",
    "title" : "keys-api"
  },
  "basePath" : "/v1",
  "schemes" : [ "https" ],
  "paths" : {
    "/pets" : {
      "get" : {
        "responses" : { },
        "security" : [ {
          "api_key" : [ ]
        } ],
        "x-amazon-apigateway-integration" : {
          "type" : "http_proxy",
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets/",
          "passthroughBehavior" : "when_no_match"
        }
      },
      "post" : {
        "responses" : { },
        "security" : [ {
          "api_key" : [ ]
        } ],
        "x-amazon-apigateway-integration" : {
          "type" : "http_proxy",
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets/",
          "passthroughBehavior" : "when_no_match"
        }
      }
    }
  },
  "securityDefinitions" : {
    "api_key" : {
      "type" : "apiKey",
      "name" : "x-api-key",
      "in" : "header"
    }
  }
}
```

OpenAPI 3.0

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "keys-api",
    "version" : "2024-03-14T20:20:12Z"
  },
  "servers" : [ {
    "url" : "{basePath}",
    "variables" : {
      "basePath" : {
        "default" : "v1"
      }
    }
  } ],
  "paths" : {
    "/pets" : {
      "get" : {
        "security" : [ {
          "api_key" : [ ]
        } ],
        "x-amazon-apigateway-integration" : {
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets/",
          "passthroughBehavior" : "when_no_match",
          "type" : "http_proxy"
        }
      },
      "post" : {
        "security" : [ {
          "api_key" : [ ]
        } ],
        "x-amazon-apigateway-integration" : {
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets/",
          "passthroughBehavior" : "when_no_match",
          "type" : "http_proxy"
        }
      }
    }
  },
  "components" : {
    "securitySchemes" : {
```

```
      "api_key" : {
        "type" : "apiKey",
        "name" : "x-api-key",
        "in" : "header"
      }
    }
  }
}
```

APIFormato del file API della chiave Gateway

APIGateway può importare API chiavi da file esterni in un formato con valori separati da virgole (CSV) e quindi associare le chiavi importate a uno o più piani di utilizzo. Il file importato deve contenere le colonne `Name` e `Key`. Per i nomi delle intestazioni di colonna non viene rilevata la distinzione tra maiuscole e minuscole e le colonne possono essere disposte in qualsiasi ordine, come mostrato nel'esempio seguente:

```
Key,name
apikey1234abcdefghij0123456789,MyFirstApiKey
```

Un `Key` valore deve essere compreso tra 20 e 128 caratteri. Un valore `Name` non può superare 1024 caratteri.

Un file API chiave può anche avere la `UsagePlanIds` colonna `DescriptionEnabled`, o, come illustrato nell'esempio seguente:

```
Name,key,description,Enabled,usageplanIds
MyFirstApiKey,apikey1234abcdefghij0123456789,An imported key,TRUE,c7y23b
```

Quando una chiave è associata a più di un piano di utilizzo, il `UsagePlanIds` valore è una stringa separata da virgole del piano di utilizzoIDs, racchiusa tra un paio di virgolette doppie o singole, come illustrato nell'esempio seguente:

```
Enabled,Name,key,UsageplanIds
true,MyFirstApiKey,apikey1234abcdefghij0123456789,"c7y23b,glvrsr"
```

Le colonne non riconosciute sono consentite ma vengono ignorate. Il valore predefinito è una stringa vuota o il valore booleano `true`.

La stessa API chiave può essere importata più volte, con la versione più recente che sovrascrive quella precedente. Due API chiavi sono identiche se hanno lo stesso `key` valore.

> **Note**
>
> Per informazioni sulle best practice da prendere in considerazione, consulta the section called "Le migliori pratiche per API chiavi e piani di utilizzo".

Documentazione per REST APIs in API Gateway

Per aiutare i clienti a comprendere e utilizzare la tuaAPI, devi documentare ilAPI. Per aiutarvi a documentare il vostroAPI, API Gateway vi consente di aggiungere e aggiornare il contenuto della guida per API le singole entità come parte integrante del processo di API sviluppo. APIGateway archivia il contenuto di origine e consente di archiviare diverse versioni della documentazione. È possibile associare una versione della documentazione a una API fase, esportare un'istantanea della documentazione specifica dello stadio in un API file Open esterno e distribuire il file come pubblicazione della documentazione.

Per documentare il proprioAPI, è possibile chiamare il APIGateway REST API, utilizzare uno dei AWS SDKs, utilizzare AWS CLIfor API Gateway o utilizzare la API console Gateway. Inoltre, è possibile importare o esportare le parti della documentazione definite in un API file Open esterno.

Per condividere API la documentazione con gli sviluppatori, puoi utilizzare un portale per sviluppatori. Per un esempio, consulta Integrating ReadMe with API Gateway to Keep Your Developer Hub aggiornato o How to Streamline API Development on Amazon API Gateway SmartBear Using SwaggerHub on the AWS Partner Network (APN) blog.

Argomenti

- Rappresentazione della API documentazione in Gateway API
- Documenta e API utilizza la console API Gateway
- Pubblica API la documentazione utilizzando la console API Gateway
- Documenta e API utilizza il API Gateway REST API
- Pubblica API la documentazione utilizzando il API Gateway REST API
- Importa API la documentazione
- Controlla l'accesso alla API documentazione in API Gateway

Rappresentazione della API documentazione in Gateway API

APILa API documentazione del gateway è costituita da singole parti della documentazione associate a API entità specifiche che includono risorseAPI, metodi, richieste, risposte, parametri del messaggio (ad esempio, path, query, header), nonché autorizzatori e modelli.

In API Gateway, una parte della documentazione è rappresentata da una DocumentationPartrisorsa. La API documentazione nel suo insieme è rappresentata dalla DocumentationPartsraccolta.

La documentazione API implica la creazione di `DocumentationPart` istanze, l'aggiunta alla `DocumentationParts` raccolta e la manutenzione delle versioni delle parti della documentazione man mano API che la documentazione si evolve.

Argomenti

- Parti della documentazione
- Versioni della documentazione

Parti della documentazione

Una DocumentationPartrisorsa è un JSON oggetto che memorizza il contenuto della documentazione applicabile a una singola entità. API Il campo `properties` contiene il contenuto della documentazione sotto forma di mappa di coppie chiave-valore. `location`La sua proprietà identifica l'APIentità associata.

La forma di una mappa dei contenuti è determinata da te, lo API sviluppatore. Il valore di una coppia chiave-valore può essere una stringa, un numero, un booleano, un oggetto o una matrice. La forma dell'oggetto `location` dipende dal tipo di entità di destinazione.

La `DocumentationPart` risorsa supporta l'ereditarietà dei contenuti: il contenuto della documentazione di un'APIentità è applicabile ai figli di tale API entità. Per ulteriori informazioni sulla definizione di entità secondarie e sull'ereditarietà del contenuto, consulta Ereditare il contenuto da un'APIentità con specifiche più generali.

Posizione di una parte della documentazione

La proprietà location di un'DocumentationPartistanza identifica un'APIentità a cui si applica il contenuto associato. L'APIentità può essere una REST API risorsa API Gateway, ad esempio Resource RestApi, Method MethodResponse, Authorizer o Model. L'entità può anche essere un parametro di messaggio, ad esempio un parametro URL path, un parametro della stringa di query, un

parametro di intestazione della richiesta o della risposta, un corpo della richiesta o della risposta o un codice di stato della risposta.

Per specificare un'APIentità, imposta l'attributo type dell'`location`oggetto in modo che sia uno di `API` `AUTHORIZERMODEL`,`RESOURCE`,`METHOD`,`PATH_PARAMETER`,`QUERY_PARAMETER`,`REQUEST_HEADER`,`REQUES` `RESPONSERESPONSE_HEADER`, o`RESPONSE_BODY`.

A seconda `type` dell'APIentità, potete specificare altri `location` attributi, tra cui metodo, nome, percorso e statusCode. Non tutti questi attributi sono validi per una determinata API entità. Ad esempio, `type`, `path`, `name` e `statusCode` sono attributi validi dell'entità `RESPONSE`, solo `type` e `path` sono attributi di posizione validi dell'entità `RESOURCE`. È un errore includere un campo non valido nel campo `location` `DocumentationPart` di una determinata API entità.

Non tutti i campi `location` validi sono obbligatori. Ad esempio, `type` è sia il `location` campo valido che quello obbligatorio di tutte le API entità. Tuttavia, `method`, `path` e `statusCode` sono attributi validi ma non obbligatori per l'entità `RESPONSE`. Se non è specificato esplicitamente, un campo `location` valido assume il valore predefinito. Il `path` valore predefinito è`/`, ad esempio, la risorsa principale di unAPI. Il valore predefinito di `method` o `statusCode` è `*`, ossia qualsiasi valore di metodo o codice di stato, rispettivamente.

Contenuto di una parte della documentazione

Il `properties` valore è codificato come JSON stringa. Il valore `properties` contiene tutte le informazioni che hai scelto per soddisfare i tuoi requisiti di documentazione. Ad esempio, la seguente è una mappa di contenuti valida:

```
{
  "info": {
    "description": "My first API with Amazon API Gateway."
  },
  "x-custom-info" : "My custom info, recognized by OpenAPI.",
  "my-info" : "My custom info not recognized by OpenAPI."
}
```

Sebbene API Gateway accetti qualsiasi JSON stringa valida come mappa del contenuto, gli attributi di contenuto vengono trattati come due categorie: quelli che possono essere riconosciuti da Open API e quelli che non possono essere riconosciuti. Nell'esempio precedente, `infodescription`, e `x-custom-info` sono riconosciuti da Open API come API oggetto, proprietà o estensione Open standard. Al contrario, non `my-info` è conforme alla specifica Open. API APIGateway propaga

gli attributi API di contenuto conformi a Open nelle definizioni di API entità dalle istanze associate. `DocumentationPart` APIGateway non propaga gli attributi di contenuto non conformi nelle definizioni delle entità. API

Per un altro esempio, ecco un `DocumentationPart` mirato per un'entità `Resource`:

```
{
    "location" : {
        "type" : "RESOURCE",
        "path": "/pets"
    },
    "properties" : {
        "summary" : "The /pets resource represents a collection of pets in PetStore.",
        "description": "... a child resource under the root...",
    }
}
```

Qui, `type` e `path` sono entrambi campi validi per l'identificazione della destinazione del tipo `RESOURCE`. Per la risorsa radice (`/`), puoi omettere il campo `path`.

```
{
    "location" : {
        "type" : "RESOURCE"
    },
    "properties" : {
        "description" : "The root resource with the default path specification."
    }
}
```

È equivalente alla seguente istanza di `DocumentationPart`:

```
{
    "location" : {
        "type" : "RESOURCE",
        "path": "/"
    },
    "properties" : {
        "description" : "The root resource with an explicit path specification"
    }
}
```

Eredita il contenuto da un'APIentità con specifiche più generali

Il valore predefinito di un `location` campo opzionale fornisce una descrizione modellata di un'APIentità. Usando il valore predefinito dell'oggetto `location`, puoi aggiungere una descrizione generale nella mappa di `properties` a un'istanza di `DocumentationPart` con questo tipo di modello `location`. APIGateway estrae gli attributi `DocumentationPart` di Open API documentation applicabili dall'APIentità generica e li inserisce in API un'entità specifica i cui `location` campi corrispondono allo `location` schema generale o al valore esatto, a meno che l'entità specifica non abbia già un'`DocumentationPart`istanza associata. Questo comportamento è noto anche come ereditarietà del contenuto da un'APIentità con specifiche più generali.

L'ereditarietà dei contenuti non si applica a determinati tipi di entità. API Per informazioni dettagliate, consulta la tabella seguente.

Quando un'APIentità corrisponde a più di uno schema `DocumentationPart` di localizzazione, erediterà la parte relativa alla documentazione con i campi di localizzazione con la precedenza e le specificità più elevate. L'ordine di precedenza è `path` > `statusCode`. Per la corrispondenza con il `path` campo, API Gateway sceglie l'entità con il valore di percorso più specifico. La seguente tabella mostra questo comportamento con alcuni esempi.

Caso	**path**	**state**	**name**	Remar
1	`/pets`	`*`	`id`	La documentazione associata a questo modello di posizione verrà ereditata dalle entità che corrispon

Caso	`path`	`state`	`name`	Remar
				dono al modello di posizione.

Caso	**path**	**state**	**name**	Remar
2	/pets	200	id	La documentazione associata a questo modello di posizione verrà ereditata dalle entità che corrispondono al modello di posizione in caso di corrispondenza con il caso 1 e il caso 2, in quanto il caso

Caso	`path`	`state`	`name`	Remar
				2 è più specifico del caso 1.

Caso	**path**	**state**	**name**	Remar
3	`/pets/petId`	`*`	`id`	La documentazione associata a questo modello di posizione verrà ereditata dalle entità che corrispondono al modello di posizione quando i casi 1, 2 e 3 sono corrispondenti, poiché il caso 3 ha una precedenz

Caso	path	state	name	Remar
				a maggiore rispetto al caso 2 ed è più specifico del caso 1.

Ecco un altro esempio per contrapporre un'istanza di `DocumentationPart` più generica con una più specifica. Il seguente messaggio di errore generale di `"Invalid request error"` viene inserito nelle API definizioni Open delle risposte di `400` errore, a meno che non venga sovrascritto.

```
{
    "location" : {
        "type" : "RESPONSE",
        "statusCode": "400"
    },
    "properties" : {
        "description" : "Invalid request error."
    }"
}
```

Con la seguente sostituzione, l'errore `400` che risponde ai metodi della risorsa `/pets` ha la descrizione `"Invalid petId specified"`.

```
{
    "location" : {
        "type" : "RESPONSE",
        "path": "/pets",
        "statusCode": "400"
    },
    "properties" : "{
```

```
        "description" : "Invalid petId specified."
    }"
}
```

Campi di posizione di valid **`DocumentationPart`**

La tabella seguente mostra i campi validi e obbligatori, nonché i valori predefiniti applicabili di una DocumentationPartrisorsa associata a un determinato tipo di entità. API

Entità API	Campi di posizione validi	Campi di posizione obbligatori	Valori dei campi predefiniti	Contenuto ereditabile
API	`{ "location": { "type": "API" }, ... }`	`type`	N/D	No
Resource (Risorsa)	`{ "location": { "type": "RESOURCE", "path": "resource_path" }, ... }`	`type`	Il valore predefinito di `path` è `/`.	No
Metodo	`{ "location": { "type": "METHOD", "path": "resource_path", "method": "http_verb" }, ...`	`type`	I valori predefiniti di `path` e `method` sono `/` e `*`, rispettivamente.	Sì, con corrispondenza di `path` per il prefisso e corrispondenza di `method` per

Entità API	Campi di posizione validi	Campi di posizione obbligatori	Valori dei campi predefiniti	Contenuto ereditabile
	`}`			qualsiasi valore.
Parametro di query	`{ "location": { "type": "QUERY_PARAMETER", "path": "resource_path ", "method": "HTTP_verb ", "name": "query_parameter_name " }, ... }`	`type`	I valori predefiniti di `path` e `method` sono `/` e `*`, rispettivamente.	Sì, con corrispondenza di `path` per il prefisso e corrispondenza di `method` per i valori esatti.
Corpo di richiesta	`{ "location": { "type": "REQUEST_BODY", "path": "resource_path ", "method": "http_verb " }, ... }`	`type`	I valori predefiniti di `path` e `method` sono `/` e `*`, rispettivamente.	Sì, con corrispondenza di `path` per il prefisso e corrispondenza di `method` per i valori esatti.

Entità API	Campi di posizione validi	Campi di posizione obbligatori	Valori dei campi predefiniti	Contenuto ereditabile
Parametro di intestazione di richiesta	`{ "location": { "type": "REQUEST_HEADER", "path": "resource_path ", "method": "HTTP_verb ", "name": "header_name " }, ... }`	`type`, `name`	I valori predefiniti di `path` e `method` sono `/` e `*`, rispettivamente.	Sì, con corrispondenza di `path` per il prefisso e corrispondenza di `method` per i valori esatti.
Parametro di percorso richiesta	`{ "location": { "type": "PATH_PARAMETER", "path": "resource/{path_parameter_name }", "method": "HTTP_verb ", "name": "path_parameter_name " }, ... }`	`type`, `name`	I valori predefiniti di `path` e `method` sono `/` e `*`, rispettivamente.	Sì, con corrispondenza di `path` per il prefisso e corrispondenza di `method` per i valori esatti.

Entità API	Campi di posizione validi	Campi di posizione obbligatori	Valori dei campi predefiniti	Contenuto ereditabile
Risposta	`{ "location": { "type": "RESPONSE", "path": "resource_path ", "method": "http_verb ", "statusCode": "status_code " }, ... }`	`type`	I valori predefiniti di `path`, `method` e `statusCode` sono `/`, `*` e `*`, rispettivamente.	Sì, con corrispondenza di `path` per il prefisso e corrispondenza di `method` e `statusCode` per i valori esatti.
Intestazione della risposta	`{ "location": { "type": "RESPONSE_HEADER", "path": "resource_path ", "method": "http_verb ", "statusCode": "status_code ", "name": "header_name " }, ... }`	`type`, `name`	I valori predefiniti di `path`, `method` e `statusCode` sono `/`, `*` e `*`, rispettivamente.	Sì, con corrispondenza di `path` per il prefisso e corrispondenza di `method` e `statusCode` per i valori esatti.

Entità API	Campi di posizione validi	Campi di posizione obbligatori	Valori dei campi predefiniti	Contenuto ereditabile
Corpo di risposta	`{ "location": { "type": "RESPONSE_BODY", "path": "resource_path ", "method": "http_verb ", "statusCode": "status_code " }, ... }`	`type`	I valori predefiniti di `path`, `method` e `statusCode` sono `/`, `*` e `*`, rispettivamente.	Sì, con corrispondenza di `path` per il prefisso e corrispondenza di `method` e `statusCode` per i valori esatti.
Authorizer	`{ "location": { "type": "AUTHORIZER", "name": "authorizer_name " }, ... }`	`type`	N/A	No
Modello	`{ "location": { "type": "MODEL", "name": "model_name " }, ... }`	`type`	N/A	No

Versioni della documentazione

Una versione della documentazione è un'istantanea della DocumentationPartsraccolta di un file API ed è contrassegnata con un identificatore di versione. La pubblicazione della documentazione di un API implica la creazione di una versione della documentazione, l'associazione a una API fase e l'esportazione di quella versione della API documentazione specifica per la fase in un file Open esterno. API In API Gateway, un'istantanea della documentazione è rappresentata come una risorsa. DocumentationVersion

Quando si aggiorna unAPI, si creano nuove versioni diAPI. In API Gateway, si mantengono tutte le versioni della documentazione utilizzando la DocumentationVersionsraccolta.

Documenta e API utilizza la console API Gateway

In questa sezione, descriviamo come creare e gestire parti della documentazione relative all'APIutilizzo della console API Gateway.

Un prerequisito per creare e modificare la documentazione di un API è che sia già stato creato ilAPI. In questa sezione, utilizziamo PetStoreAPIcome esempio. Per crearne una API utilizzando la console API Gateway, segui le istruzioni riportate inTutorial: creazione di un'API REST mediante l'importazione di un esempio.

Argomenti

- Documentare l'entità API
- Documentare un'entità RESOURCE
- Documentare un'entità METHOD
- Documentare un'entità QUERY_PARAMETER
- Documentare un'entità PATH_PARAMETER
- Documentare un'entità REQUEST_HEADER
- Documentare un'entità REQUEST_BODY
- Documentare un'entità RESPONSE
- Documentare un'entità RESPONSE_HEADER
- Documentare un'entità RESPONSE_BODY
- Documentare un'entità MODEL
- Documentare un'entità AUTHORIZER

Documentare l'entità **API**

Per aggiungere una nuova parte della documentazione per l'entità `API`, procedi come segue:

1. Nel riquadro di navigazione principale scegli Documentazione, quindi seleziona Crea parte della documentazione.
2. Per Tipo di documentazione, seleziona API.

   Se una parte della documentazione non è stata creata per l'`API`, viene visualizzato l'editor della mappa di `properties` della parte della documentazione. Inserisci la seguente mappa di `properties` nell'editor di testo.

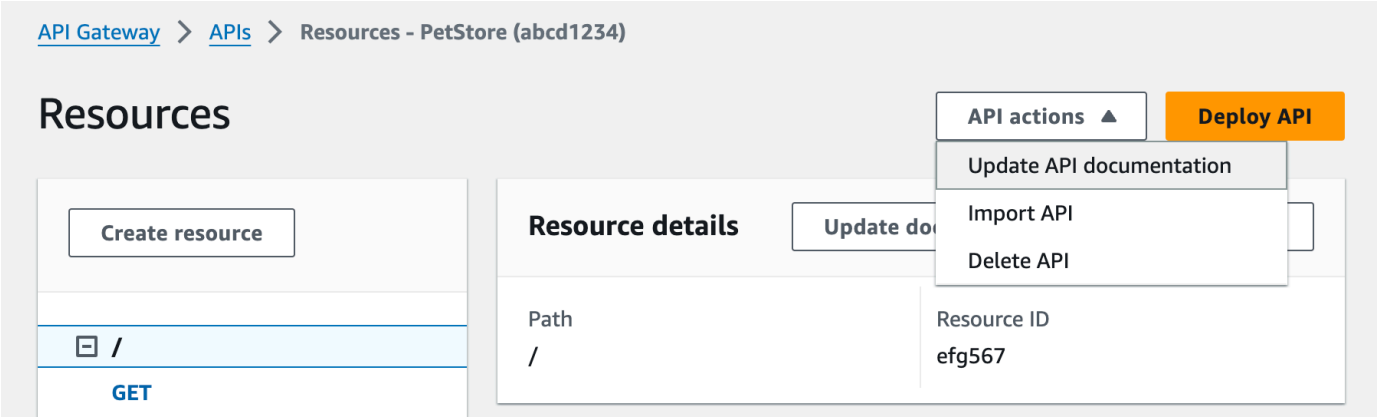
   ```
   {
     "info": {
       "description": "Your first API Gateway API.",
       "contact": {
           "name": "John Doe",
           "email": "john.doe@api.com"
       }
     }
   }
   ```

   > **Note**
   >
   > Non è necessario codificare la `properties` mappa in una JSON stringa. La console API Gateway stringe l'oggetto al JSON posto tuo.

3. Scegli Crea parte della documentazione.

Per aggiungere una nuova parte della documentazione per l'entità `API` nel riquadro Risorse, procedi come segue:

1. Nel riquadro di navigazione principale scegli Risorse.
2. Scegli il menu APIdelle azioni, quindi scegli Aggiorna documentazione. API



Per modificare una parte della documentazione esistente, procedi come segue:

1. Nel riquadro Documentazione scegli la scheda Risorse e metodi.
2. Seleziona il nome del tuoAPI, quindi sulla API scheda scegli Modifica.

Documentare un'entità **`RESOURCE`**

Per aggiungere una nuova parte della documentazione per un'entità `RESOURCE`, procedi come segue:

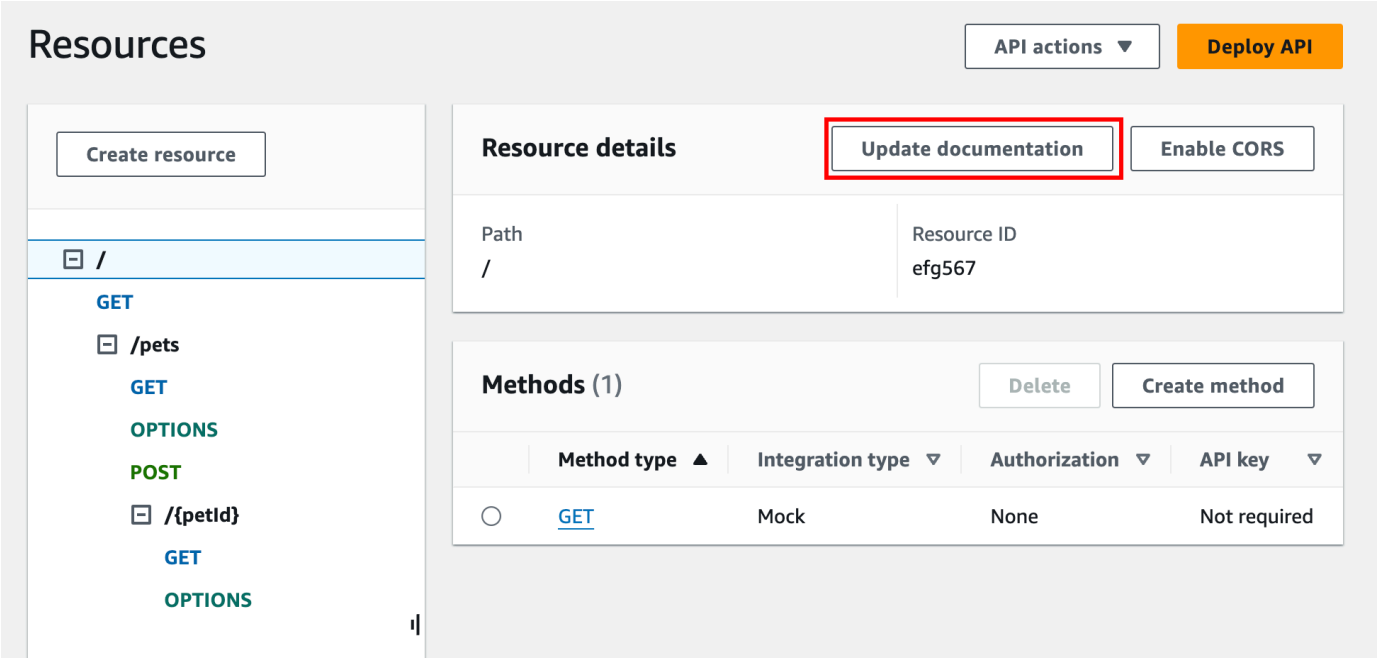
1. Nel riquadro di navigazione principale scegli Documentazione, quindi seleziona Crea parte della documentazione.
2. Per Tipo di documentazione seleziona Risorsa.
3. Per Percorso inserisci un percorso.
4. Immetti una descrizione nell'editor di testo, ad esempio:

```
{
    "description": "The PetStore's root resource."
}
```

5. Scegli Crea parte della documentazione. È possibile creare la documentazione per una risorsa non elencata.
6. Se necessario, ripeti queste fasi per aggiungere o modificare un'altra parte della documentazione.

Per aggiungere una nuova parte della documentazione per un'entità `RESOURCE` nel riquadro Risorse, procedi come segue:

1. Nel riquadro di navigazione principale scegli Risorse.
2. Scegli la risorsa, quindi seleziona Aggiorna documentazione.



Per modificare una parte della documentazione esistente, procedi come segue:

1. Nel riquadro Documentazione scegli la scheda Risorse e metodi.
2. Seleziona la risorsa contenente la parte della documentazione, quindi scegli Modifica.

Documentare un'entità `METHOD`

Per aggiungere una nuova parte della documentazione per un'entità `METHOD`, procedi come segue:

1. Nel riquadro di navigazione principale scegli Documentazione, quindi seleziona Crea parte della documentazione.
2. Per Tipo di documentazione seleziona Metodo.
3. Per Percorso inserisci un percorso.
4. Per Metodo, seleziona un HTTP verbo.
5. Immetti una descrizione nell'editor di testo, ad esempio:

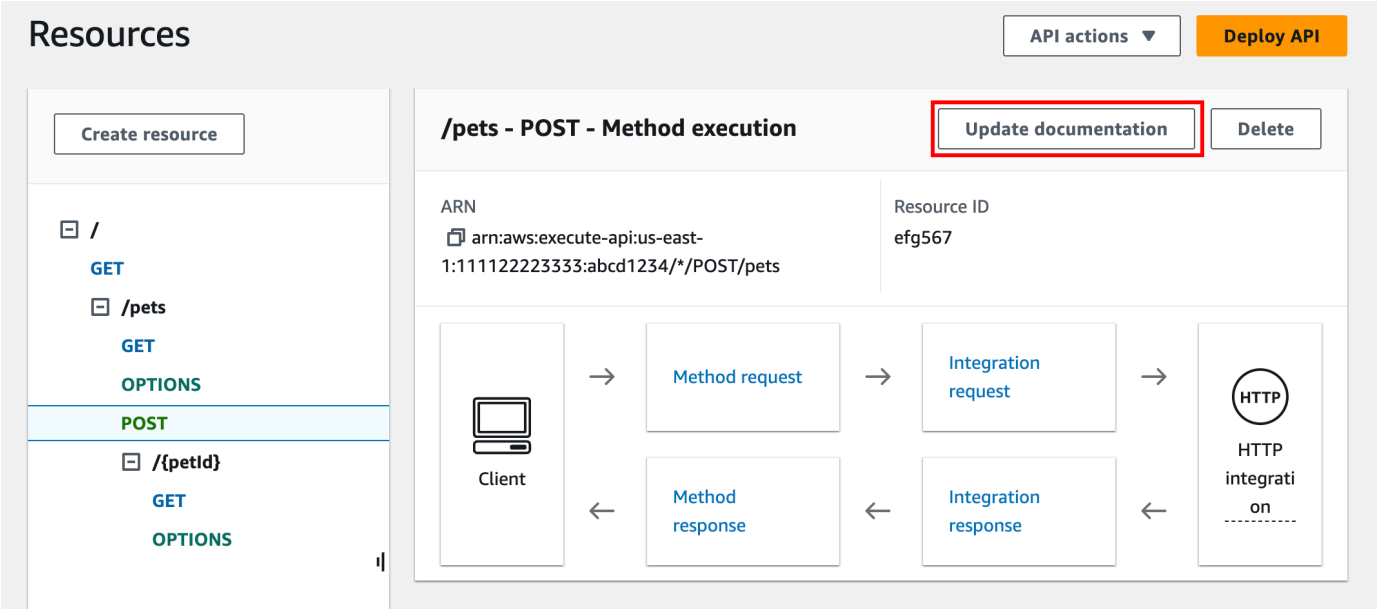
```
{
  "tags" : [ "pets" ],
```

```
  "summary" : "List all pets"
}
```

6. Scegli Crea parte della documentazione. È possibile creare la documentazione per un metodo non elencato.
7. Se necessario, ripeti queste fasi per aggiungere o modificare un'altra parte della documentazione.

Per aggiungere una nuova parte della documentazione per un'entità `METHOD` nel riquadro Risorse, procedi come segue:

1. Nel riquadro di navigazione principale scegli Risorse.
2. Scegli il metodo, quindi seleziona Aggiorna documentazione.



Per modificare una parte della documentazione esistente, procedi come segue:

1. Nel riquadro Documentazione scegli la scheda Risorse e metodi.
2. È possibile selezionare il metodo o la risorsa contenente il metodo, quindi utilizzare la barra di ricerca per trovare e scegliere la parte della documentazione.
3. Scegli Modifica.

Documentare un'entità **QUERY_PARAMETER**

Per aggiungere una nuova parte della documentazione per un'entità `QUERY_PARAMETER`, procedi come segue:

1. Nel riquadro di navigazione principale scegli Documentazione, quindi seleziona Crea parte della documentazione.
2. Per Tipo di documentazione seleziona Parametro di query.
3. Per Percorso inserisci un percorso.
4. Per Metodo, seleziona un HTTP verbo.
5. In Nome, immetti un nome.
6. Immetti una descrizione nell'editor di testo.
7. Scegli Crea parte della documentazione. È possibile creare la documentazione per un parametro di query non elencato.
8. Se necessario, ripeti queste fasi per aggiungere o modificare un'altra parte della documentazione.

Per modificare una parte della documentazione esistente, procedi come segue:

1. Nel riquadro Documentazione scegli la scheda Risorse e metodi.
2. È possibile selezionare il parametro di query o la risorsa contenente il parametro di query, quindi utilizzare la barra di ricerca per trovare e scegliere la parte della documentazione.
3. Scegli Modifica.

Documentare un'entità **PATH_PARAMETER**

Per aggiungere una nuova parte della documentazione per un'entità `PATH_PARAMETER`, procedi come segue:

1. Nel riquadro di navigazione principale scegli Documentazione, quindi seleziona Crea parte della documentazione.
2. Per Tipo di documentazione seleziona Parametro di percorso.
3. Per Percorso inserisci un percorso.
4. Per Metodo, selezionate un HTTP verbo.

5. In Nome, immetti un nome.
6. Immetti una descrizione nell'editor di testo.
7. Scegli Crea parte della documentazione. È possibile creare la documentazione per un parametro di percorso non elencato.
8. Se necessario, ripeti queste fasi per aggiungere o modificare un'altra parte della documentazione.

Per modificare una parte della documentazione esistente, procedi come segue:

1. Nel riquadro Documentazione scegli la scheda Risorse e metodi.
2. È possibile selezionare il parametro di percorso o la risorsa contenente il parametro di percorso, quindi utilizzare la barra di ricerca per trovare e scegliere la parte della documentazione.
3. Scegli Modifica.

Documentare un'entità **`REQUEST_HEADER`**

Per aggiungere una nuova parte della documentazione per un'entità `REQUEST_HEADER`, procedi come segue:

1. Nel riquadro di navigazione principale scegli Documentazione, quindi seleziona Crea parte della documentazione.
2. Per Tipo di documentazione seleziona Intestazione della richiesta.
3. Per Percorso inserisci il percorso dell'intestazione della richiesta.
4. Per Metodo, selezionate un HTTP verbo.
5. In Nome, immetti un nome.
6. Immetti una descrizione nell'editor di testo.
7. Scegli Crea parte della documentazione. È possibile creare la documentazione per un'intestazione della richiesta non elencata.
8. Se necessario, ripeti queste fasi per aggiungere o modificare un'altra parte della documentazione.

Per modificare una parte della documentazione esistente, procedi come segue:

1. Nel riquadro Documentazione scegli la scheda Risorse e metodi.

2. È possibile selezionare l'intestazione della richiesta o la risorsa contenente l'intestazione della richiesta, quindi utilizzare la barra di ricerca per trovare e scegliere la parte della documentazione.
3. Scegli Modifica.

Documentare un'entità **`REQUEST_BODY`**

Per aggiungere una nuova parte della documentazione per un'entità `REQUEST_BODY`, procedi come segue:

1. Nel riquadro di navigazione principale scegli Documentazione, quindi seleziona Crea parte della documentazione.
2. Per Tipo di documentazione seleziona Corpo della richiesta.
3. Per Percorso inserisci il percorso del corpo della richiesta.
4. Per Metodo, selezionate un HTTP verbo.
5. Immetti una descrizione nell'editor di testo.
6. Scegli Crea parte della documentazione. È possibile creare la documentazione per un corpo della richiesta non elencato.
7. Se necessario, ripeti queste fasi per aggiungere o modificare un'altra parte della documentazione.

Per modificare una parte della documentazione esistente, procedi come segue:

1. Nel riquadro Documentazione scegli la scheda Risorse e metodi.
2. È possibile selezionare il corpo della richiesta o la risorsa contenente il corpo della richiesta, quindi utilizzare la barra di ricerca per trovare e scegliere la parte della documentazione.
3. Scegli Modifica.

Documentare un'entità **`RESPONSE`**

Per aggiungere una nuova parte della documentazione per un'entità `RESPONSE`, procedi come segue:

1. Nel riquadro di navigazione principale scegli Documentazione, quindi seleziona Crea parte della documentazione.
2. Per Tipo di documentazione seleziona Risposta (codice di stato).

3. Per Percorso inserisci un percorso per la risposta.
4. Per Metodo, selezionate un HTTP verbo.
5. Per Codice di stato, inserisci un codice di HTTP stato.
6. Immetti una descrizione nell'editor di testo.
7. Scegli Crea parte della documentazione. È possibile creare la documentazione per un codice di stato della risposta non elencato.
8. Se necessario, ripeti queste fasi per aggiungere o modificare un'altra parte della documentazione.

Per modificare una parte della documentazione esistente, procedi come segue:

1. Nel riquadro Documentazione scegli la scheda Risorse e metodi.
2. È possibile selezionare il codice di stato della risposta o la risorsa contenente il codice di stato della risposta, quindi utilizzare la barra di ricerca per trovare e scegliere la parte della documentazione.
3. Scegli Modifica.

Documentare un'entità **`RESPONSE_HEADER`**

Per aggiungere una nuova parte della documentazione per un'entità `RESPONSE_HEADER`, procedi come segue:

1. Nel riquadro di navigazione principale scegli Documentazione, quindi seleziona Crea parte della documentazione.
2. Per Tipo di documentazione seleziona Intestazione della risposta.
3. Per Percorso inserisci un percorso per l'intestazione della risposta.
4. Per Metodo, selezionate un HTTP verbo.
5. Per Codice di stato, inserisci un codice di HTTP stato.
6. Immetti una descrizione nell'editor di testo.
7. Scegli Crea parte della documentazione. È possibile creare la documentazione per un'intestazione della risposta non elencata.
8. Se necessario, ripeti queste fasi per aggiungere o modificare un'altra parte della documentazione.

Per modificare una parte della documentazione esistente, procedi come segue:

1. Nel riquadro Documentazione scegli la scheda Risorse e metodi.
2. È possibile selezionare l'intestazione della risposta o la risorsa contenente l'intestazione della risposta, quindi utilizzare la barra di ricerca per trovare e scegliere la parte della documentazione.
3. Scegli Modifica.

Documentare un'entità **`RESPONSE_BODY`**

Per aggiungere una nuova parte della documentazione per un'entità `RESPONSE_BODY`, procedi come segue:

1. Nel riquadro di navigazione principale scegli Documentazione, quindi seleziona Crea parte della documentazione.
2. Per Tipo di documentazione seleziona Corpo della risposta.
3. Per Percorso inserisci un percorso per il corpo della risposta.
4. Per Metodo, selezionate un HTTP verbo.
5. Per Codice di stato, inserisci un codice di HTTP stato.
6. Immetti una descrizione nell'editor di testo.
7. Scegli Crea parte della documentazione. È possibile creare la documentazione per un corpo della risposta non elencato.
8. Se necessario, ripeti queste fasi per aggiungere o modificare un'altra parte della documentazione.

Per modificare una parte della documentazione esistente, procedi come segue:

1. Nel riquadro Documentazione scegli la scheda Risorse e metodi.
2. È possibile selezionare il corpo della risposta o la risorsa contenente il corpo della risposta, quindi utilizzare la barra di ricerca per trovare e scegliere la parte della documentazione.
3. Scegli Modifica.

Documentare un'entità **`MODEL`**

La documentazione di un'entità `MODEL` comporta la creazione e la gestione delle istanze di `DocumentPart` per il modello e gli elementi `properties` del modello. Ad esempio, per impostazione API predefinita, il `Error` modello fornito con every ha la seguente definizione di schema,

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "Error Schema",
  "type" : "object",
  "properties" : {
    "message" : { "type" : "string" }
  }
}
```

e richiede due istanze di `DocumentationPart`, una per `Model` e l'altra per la relativa proprietà `message`:

```
{
  "location": {
    "type": "MODEL",
    "name": "Error"
  },
  "properties": {
    "title": "Error Schema",
    "description": "A description of the Error model"
  }
}
```

e

```
{
  "location": {
    "type": "MODEL",
    "name": "Error.message"
  },
  "properties": {
    "description": "An error message."
  }
}
```

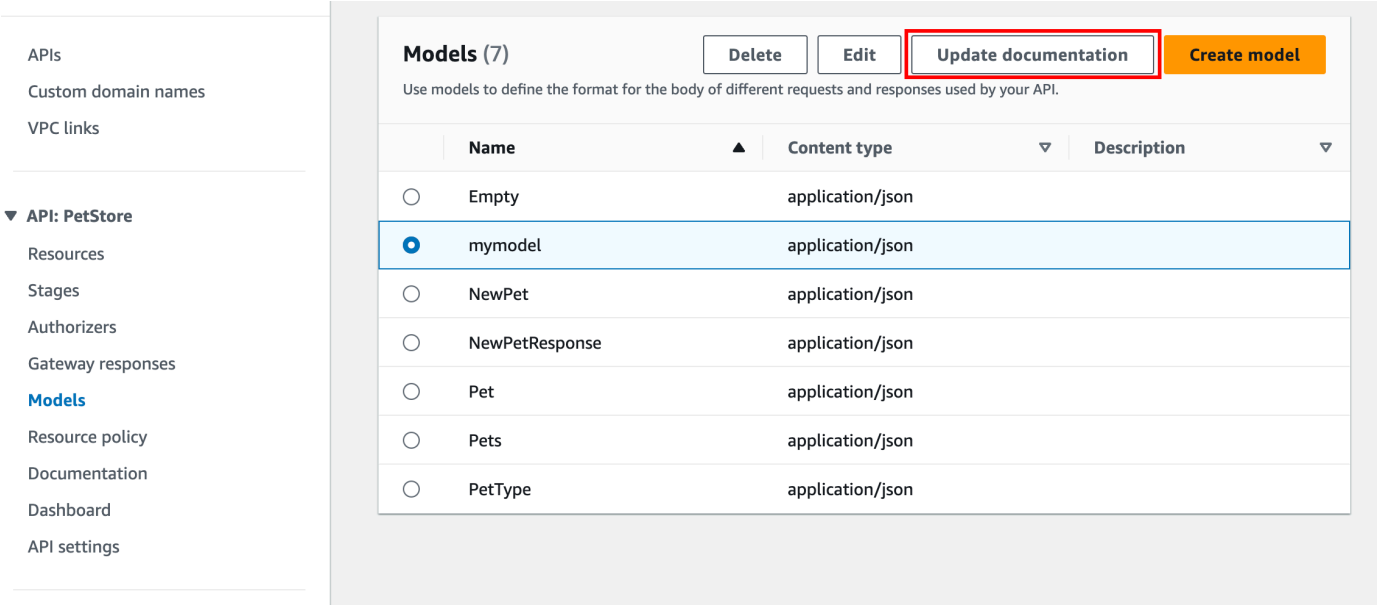
Quando API viene esportato, le `DocumentationPart` sue proprietà sostituiranno i valori dello schema originale.

Per aggiungere una nuova parte della documentazione per un'entità `MODEL`, procedi come segue:

1. Nel riquadro di navigazione principale scegli Documentazione, quindi seleziona Crea parte della documentazione.
2. Per Tipo di documentazione seleziona Modello.
3. Per Nome inserisci un nome per il processo.
4. Immetti una descrizione nell'editor di testo.
5. Scegli Crea parte della documentazione. È possibile creare la documentazione per modelli non elencati.
6. Se necessario, ripeti queste fasi per aggiungere o modificare una parte della documentazione per altri modelli.

Per aggiungere una nuova parte della documentazione per un'entità `MODEL` nel riquadro Modelli, procedi come segue:

1. Nel riquadro di navigazione principale seleziona Modelli.
2. Scegli il modello, quindi seleziona Aggiorna documentazione.



Per modificare una parte della documentazione esistente, procedi come segue:

1. Nel riquadro Documentazione scegli la scheda Modelli.
2. Utilizza la barra di ricerca o seleziona il modello, quindi scegli Modifica.

Documentare un'entità **`AUTHORIZER`**

Per aggiungere una nuova parte della documentazione per un'entità `AUTHORIZER`, procedi come segue:

1. Nel riquadro di navigazione principale scegli Documentazione, quindi seleziona Crea parte della documentazione.
2. Per Tipo di documentazione seleziona Sistema di autorizzazione.
3. Per Nome immetti il nome del sistema di autorizzazione.
4. Immetti una descrizione nell'editor di testo. Specifica un valore per il campo `location` valido per il sistema di autorizzazione.
5. Scegli Crea parte della documentazione. È possibile creare la documentazione per sistemi di autorizzazione non elencati.
6. Se necessario, ripeti queste fasi per aggiungere o modificare una parte della documentazione per altre autorizzazioni.

Per modificare una parte della documentazione esistente, procedi come segue:

1. Nel riquadro Documentazione scegli la scheda Sistemi di autorizzazione.
2. Utilizza la barra di ricerca o seleziona il sistema di autorizzazione, quindi scegli Modifica.

Pubblica API la documentazione utilizzando la console API Gateway

La procedura seguente mostra come pubblicare una versione della documentazione.

Per pubblicare una versione della documentazione utilizzando la console API Gateway

1. Nel riquadro di navigazione principale scegli Documentazione.
2. Seleziona Pubblica documentazione.
3. Configura la pubblicazione:
   a. In Fase, seleziona una fase.
   b. Per Versione inserisci un identificatore di versione, ad esempio `1.0.0`.

c. (Facoltativo) In Description (Descrizione), immettere una descrizione.

4. Seleziona Publish (Pubblica).

È ora possibile procedere al download della documentazione pubblicata esportando la documentazione in un API file Open esterno. Per ulteriori informazioni, consulta the section called “Esporta un REST API”.

Documenta e API utilizza il API Gateway REST API

In questa sezione, descriviamo come creare e gestire parti della documentazione relative all'APIutilizzo del API Gateway RESTAPI.

Prima di creare e modificare la documentazione di unAPI, crea innanzitutto ilAPI. In questa sezione, utilizziamo PetStoreAPIcome esempio. Per crearne una API utilizzando la console API Gateway, segui le istruzioni riportate inTutorial: creazione di un'API REST mediante l'importazione di un esempio.

Argomenti

- Documentare l'entità API
- Documentare un'entità RESOURCE
- Documentare un'entità METHOD
- Documentare un'entità QUERY_PARAMETER
- Documentare un'entità PATH_PARAMETER
- Documentare un'entità REQUEST_BODY
- Documentare un'entità REQUEST_HEADER
- Documentare un'entità RESPONSE
- Documentare un'entità RESPONSE_HEADER
- Documentare un'entità AUTHORIZER
- Documentare un'entità MODEL
- Aggiornare le parti della documentazione
- Elencare le parti della documentazione

Documentare l'entità **API**

Per aggiungere la documentazione per un API, aggiungi una DocumentationPartrisorsa per l'APIentità:

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "API"
    },
    "properties": "{\n\t\"info\": {\n\t\t\"description\" : \"Your first API with Amazon
 API Gateway.\"\n\t}\n}"
}
```

Se con esito positivo, l'operazione restituisce una risposta `201 Created` contenente la nuova istanza `DocumentationPart` creata nel payload. Ad esempio:

```
{
  ...
  "id": "s2e5xf",
  "location": {
    "path": null,
    "method": null,
    "name": null,
    "statusCode": null,
    "type": "API"
  },
  "properties": "{\n\t\"info\": {\n\t\t\"description\" : \"Your first API with Amazon
 API Gateway.\"\n\t}\n}"
}
```

Se la documentazione è già stata aggiunta, viene restituita una risposta `409 Conflict` contenente il messaggio di errore `Documentation part already exists for the specified location: type 'API'.`". In questo caso è necessario eseguire l'operazione documentationpart:update.

```
PATCH /restapis/4wk1k4onj3/documentation/parts/part_id HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "/properties",
    "value" : "{\n\t\"info\": {\n\t\t\"description\" : \"Your first API with Amazon API
 Gateway.\"\n\t}\n}"
  } ]
}
```

La risposta con esito positivo restituisce un codice di stato `200 OK` con un payload che include l'istanza `DocumentationPart` aggiornata nel payload.

Documentare un'entità **`RESOURCE`**

Per aggiungere la documentazione per la risorsa principale di unAPI, aggiungi una DocumentationPartrisorsa destinata alla risorsa Resource corrispondente:

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
       "type" : "RESOURCE",
    },
    "properties" : "{\n\t\"description\" : \"The PetStore root resource.\"\n}"
}
```

Se con esito positivo, l'operazione restituisce una risposta `201 Created` contenente la nuova istanza `DocumentationPart` creata nel payload. Ad esempio:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/p76vqo"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/p76vqo"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/p76vqo"
    }
  },
  "id": "p76vqo",
  "location": {
    "path": "/",
    "method": null,
    "name": null,
    "statusCode": null,
    "type": "RESOURCE"
  },
  "properties": "{\n\t\"description\" : \"The PetStore root resource.\"\n}"
}
```

Quando il percorso della risorsa non è specificato, si presume che la risorsa sia la risorsa radice. Puoi aggiungere `"path": "/"` a `properties` per rendere esplicita la specifica.

Per creare la documentazione per una risorsa figlia di anAPI, aggiungi una DocumentationPartrisorsa destinata alla risorsa di risorsa corrispondente:

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret
```

```
{
    "location" : {
         "type" : "RESOURCE",
         "path" : "/pets"
    },
    "properties": "{\n\t\"description\" : \"A child resource under the root of
 PetStore.\"\n}"
}
```

Se con esito positivo, l'operazione restituisce una risposta `201 Created` contenente la nuova istanza `DocumentationPart` creata nel payload. Per esempio:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/qcht86"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/qcht86"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/qcht86"
    }
  },
  "id": "qcht86",
  "location": {
    "path": "/pets",
    "method": null,
    "name": null,
    "statusCode": null,
    "type": "RESOURCE"
  },
  "properties": "{\n\t\"description\" : \"A child resource under the root of PetStore.
\"\n}"
}
```

Per aggiungere la documentazione per una risorsa secondaria specificata da un parametro path, aggiungi una DocumentationPartrisorsa destinata alla risorsa Resource:

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "RESOURCE",
         "path" : "/pets/{petId}"
    },
    "properties": "{\n\t\"description\" : \"A child resource specified by the petId
 path parameter.\"\n}"
}
```

Se con esito positivo, l'operazione restituisce una risposta `201 Created` contenente la nuova istanza `DocumentationPart` creata nel payload. Per esempio:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/k6fpwb"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/k6fpwb"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/k6fpwb"
    }
  },
  "id": "k6fpwb",
  "location": {
```

```
    "path": "/pets/{petId}",
    "method": null,
    "name": null,
    "statusCode": null,
    "type": "RESOURCE"
  },
  "properties": "{\n\t\"description\" : \"A child resource specified by the petId path
 parameter.\"\n}"
}
```

> **Note**
>
> L'DocumentationPartistanza di un'RESOURCEentità non può essere ereditata da nessuna delle sue risorse secondarie.

Documentare un'entità **METHOD**

Per aggiungere la documentazione per un metodo di anAPI, aggiungi una DocumentationPartrisorsa destinata alla risorsa Method corrispondente:

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "METHOD",
         "path" : "/pets",
         "method" : "GET"
    },
    "properties": "{\n\t\"summary\" : \"List all pets.\"\n}"
}
```

Se con esito positivo, l'operazione restituisce una risposta `201 Created` contenente la nuova istanza `DocumentationPart` creata nel payload. Ad esempio:

```
{
```

```
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    }
  },
  "id": "o64jbj",
  "location": {
    "path": "/pets",
    "method": "GET",
    "name": null,
    "statusCode": null,
    "type": "METHOD"
  },
  "properties": "{\n\t\"summary\" : \"List all pets.\"\n}"
}
```

Se con esito positivo, l'operazione restituisce una risposta `201 Created` contenente la nuova istanza `DocumentationPart` creata nel payload. Ad esempio:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    },
    "documentationpart:delete": {
```

```
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    }
  },
  "id": "o64jbj",
  "location": {
    "path": "/pets",
    "method": "GET",
    "name": null,
    "statusCode": null,
    "type": "METHOD"
  },
  "properties": "{\n\t\"summary\" : \"List all pets.\"\n}"
}
```

Se il campo `location.method` non è specificato nella richiesta precedente, si presume che sia il metodo ANY rappresentato da un carattere jolly `*`.

Per aggiornare il contenuto della documentazione di un'entità `METHOD`, chiamare l'operazione documentationpart:update, fornendo una nuova mappa di `properties`:

```
PATCH /restapis/4wk1k4onj3/documentation/parts/part_id HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "/properties",
    "value" : "{\n\t\"tags\" : [ \"pets\" ], \n\t\"summary\" : \"List all pets.\"\n}"
  } ]
}
```

La risposta con esito positivo restituisce un codice di stato `200 OK` con un payload che include l'istanza `DocumentationPart` aggiornata nel payload. Ad esempio:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    }
  },
  "id": "o64jbj",
  "location": {
    "path": "/pets",
    "method": "GET",
    "name": null,
    "statusCode": null,
    "type": "METHOD"
  },
  "properties": "{\n\t\"tags\" : [ \"pets\" ], \n\t\"summary\" : \"List all pets.\"\n}"
}
```

Documentare un'entità **QUERY_PARAMETER**

Per aggiungere la documentazione per un parametro di query di richiesta, aggiungete una DocumentationPartrisorsa mirata al `QUERY_PARAMETER` tipo, con i campi validi di `path` and`name`.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
```

```
    "location" : {
         "type" : "QUERY_PARAMETER",
         "path" : "/pets",
         "method" : "GET",
         "name" : "page"
    },
    "properties": "{\n\t\"description\" : \"Page number of results to return.\"\n}"
}
```

Se con esito positivo, l'operazione restituisce una risposta `201 Created` contenente la nuova istanza `DocumentationPart` creata nel payload. Ad esempio:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/h9ht5w"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/h9ht5w"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/h9ht5w"
    }
  },
  "id": "h9ht5w",
  "location": {
    "path": "/pets",
    "method": "GET",
    "name": "page",
    "statusCode": null,
    "type": "QUERY_PARAMETER"
  },
  "properties": "{\n\t\"description\" : \"Page number of results to return.\"\n}"
}
```

La mappa di `properties` della parte della documentazione di un'entità `QUERY_PARAMETER` può essere ereditata da uno delle relative entità figlio `QUERY_PARAMETER`. Ad esempio se aggiungi

una risorsa `treats` dopo `/pets/{petId}`, abiliti il metodo `GET` su `/pets/{petId}/treats` ed esponi il parametro di query `page`, questo parametro di query figlio eredita la mappa di `DocumentationPart` di `properties` dal parametro di query con nome simile del metodo `GET /pets`, a meno che non aggiungi esplicitamente una risorsa `DocumentationPart` al parametro di query `page` del metodo `GET /pets/{petId}/treats`.

Documentare un'entità **`PATH_PARAMETER`**

Per aggiungere la documentazione per un parametro di percorso, aggiungi una DocumentationPartrisorsa per l'PATH_PARAMETERentità.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "PATH_PARAMETER",
         "path" : "/pets/{petId}",
         "method" : "*",
         "name" : "petId"
    },
    "properties": "{\n\t\"description\" : \"The id of the pet to retrieve.\"\n}"
}
```

Se con esito positivo, l'operazione restituisce una risposta `201 Created` contenente la nuova istanza `DocumentationPart` creata nel payload. Ad esempio:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/ckpgog"
```

```
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/ckpgog"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/ckpgog"
    }
  },
  "id": "ckpgog",
  "location": {
    "path": "/pets/{petId}",
    "method": "*",
    "name": "petId",
    "statusCode": null,
    "type": "PATH_PARAMETER"
  },
  "properties": "{\n  \"description\" : \"The id of the pet to retrieve\"\n}"
}
```

Documentare un'entità **REQUEST_BODY**

Per aggiungere documentazione per un corpo della richiesta, aggiungi una DocumentationPartrisorsa per il corpo della richiesta.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "REQUEST_BODY",
         "path" : "/pets",
         "method" : "POST"
    },
    "properties": "{\n\t\"description\" : \"A Pet object to be added to PetStore.\"\n}"
}
```

Se con esito positivo, l'operazione restituisce una risposta `201 Created` contenente la nuova istanza `DocumentationPart` creata nel payload. Ad esempio:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/kgmfr1"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/kgmfr1"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/kgmfr1"
    }
  },
  "id": "kgmfr1",
  "location": {
    "path": "/pets",
    "method": "POST",
    "name": null,
    "statusCode": null,
    "type": "REQUEST_BODY"
  },
  "properties": "{\n\t\"description\" : \"A Pet object to be added to PetStore.\"\n}"
}
```

Documentare un'entità **REQUEST_HEADER**

Per aggiungere la documentazione per l'intestazione di una richiesta, aggiungi una DocumentationPartrisorsa per l'intestazione della richiesta.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
```

```
    "location" : {
         "type" : "REQUEST_HEADER",
         "path" : "/pets",
         "method" : "GET",
         "name" : "x-my-token"
    },
    "properties": "{\n\t\"description\" : \"A custom token used to authorization the
 method invocation.\"\n}"
}
```

Se con esito positivo, l'operazione restituisce una risposta `201 Created` contenente la nuova istanza `DocumentationPart` creata nel payload. Ad esempio:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/h0m3uf"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/h0m3uf"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/h0m3uf"
    }
  },
  "id": "h0m3uf",
  "location": {
    "path": "/pets",
    "method": "GET",
    "name": "x-my-token",
    "statusCode": null,
    "type": "REQUEST_HEADER"
  },
  "properties": "{\n\t\"description\" : \"A custom token used to authorization the
 method invocation.\"\n}"
}
```

Documentare un'entità `RESPONSE`

Per aggiungere la documentazione per una risposta a un codice di stato, aggiungi una DocumentationPartrisorsa destinata alla risorsa corrispondente MethodResponse.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location": {
      "path": "/",
      "method": "*",
      "name": null,
      "statusCode": "200",
      "type": "RESPONSE"
    },
    "properties": "{\n  \"description\" : \"Successful operation.\"\n}"
}
```

Se con esito positivo, l'operazione restituisce una risposta `201 Created` contenente la nuova istanza `DocumentationPart` creata nel payload. Ad esempio:

```
{
    "_links": {
        "self": {
            "href": "/restapis/4wk1k4onj3/documentation/parts/lattew"
        },
        "documentationpart:delete": {
            "href": "/restapis/4wk1k4onj3/documentation/parts/lattew"
        },
        "documentationpart:update": {
            "href": "/restapis/4wk1k4onj3/documentation/parts/lattew"
        }
    },
    "id": "lattew",
    "location": {
        "path": "/",
        "method": "*",
```

```
        "name": null,
        "statusCode": "200",
        "type": "RESPONSE"
    },
    "properties": "{\n  \"description\" : \"Successful operation.\"\n}"
}
```

Documentare un'entità **`RESPONSE_HEADER`**

Per aggiungere la documentazione per un'intestazione di risposta, aggiungi una DocumentationPartrisorsa per l'intestazione della risposta.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

  "location": {
    "path": "/",
    "method": "GET",
    "name": "Content-Type",
    "statusCode": "200",
    "type": "RESPONSE_HEADER"
  },
  "properties": "{\n  \"description\" : \"Media type of request\"\n}"
```

Se con esito positivo, l'operazione restituisce una risposta `201 Created` contenente la nuova istanza `DocumentationPart` creata nel payload. Per esempio:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/fev7j7"
    },
```

```
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/fev7j7"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/fev7j7"
    }
  },
  "id": "fev7j7",
  "location": {
    "path": "/",
    "method": "GET",
    "name": "Content-Type",
    "statusCode": "200",
    "type": "RESPONSE_HEADER"
  },
  "properties": "{\n  \"description\" : \"Media type of request\"\n}"
}
```

La documentazione di questa intestazione di `Content-Type` risposta è la documentazione predefinita per le `Content-Type` intestazioni di tutte le risposte di. API

Documentare un'entità **`AUTHORIZER`**

Per aggiungere documentazione per un API autorizzatore, aggiungi una DocumentationPartrisorsa destinata all'autorizzatore specificato.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "AUTHORIZER",
         "name" : "myAuthorizer"
    },
    "properties": "{\n\t\"description\" : \"Authorizes invocations of configured
 methods.\"\n}"
}
```

Se con esito positivo, l'operazione restituisce una risposta `201 Created` contenente la nuova istanza `DocumentationPart` creata nel payload. Per esempio:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/pw3qw3"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/pw3qw3"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/pw3qw3"
    }
  },
  "id": "pw3qw3",
  "location": {
    "path": null,
    "method": null,
    "name": "myAuthorizer",
    "statusCode": null,
    "type": "AUTHORIZER"
  },
  "properties": "{\n\t\"description\" : \"Authorizes invocations of configured methods.
\"\n}"
}
```

> **Note**
>
> L'DocumentationPartistanza di un'`AUTHORIZER`Entità non può essere ereditata da nessuna delle sue risorse secondarie.

Documentare un'entità **`MODEL`**

La documentazione di un'entità `MODEL` comporta la creazione e la gestione delle istanze di `DocumentPart` per il modello e gli elementi `properties` del modello. Ad esempio, per impostazione API predefinita, il `Error` modello fornito con ogni ha la seguente definizione di schema,

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "Error Schema",
  "type" : "object",
  "properties" : {
    "message" : { "type" : "string" }
  }
}
```

e richiede due istanze di `DocumentationPart`, una per `Model` e l'altra per la relativa proprietà `message`:

```
{
  "location": {
    "type": "MODEL",
    "name": "Error"
  },
  "properties": {
    "title": "Error Schema",
    "description": "A description of the Error model"
  }
}
```

e

```
{
  "location": {
    "type": "MODEL",
    "name": "Error.message"
  },
  "properties": {
    "description": "An error message."
  }
}
```

Quando API viene esportato, le `DocumentationPart` sue proprietà sostituiranno i valori dello schema originale.

Per aggiungere documentazione per un API modello, aggiungi una DocumentationPartrisorsa destinata al modello specificato.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "MODEL",
         "name" : "Pet"
    },
    "properties": "{\n\t\"description\" : \"Data structure of a Pet object.\"\n}"
}
```

Se con esito positivo, l'operazione restituisce una risposta `201 Created` contenente la nuova istanza `DocumentationPart` creata nel payload. Per esempio:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/lkn4uq"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/lkn4uq"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/lkn4uq"
    }
  },
```

```
  "id": "lkn4uq",
  "location": {
    "path": null,
    "method": null,
    "name": "Pet",
    "statusCode": null,
    "type": "MODEL"
  },
  "properties": "{\n\t\"description\" : \"Data structure of a Pet object.\"\n}"
}
```

Ripetete lo stesso passaggio per creare un' DocumentationPart istanza per una qualsiasi delle proprietà del modello.

> **Note**
>
> L'DocumentationPartistanza di un'`MODEL`entità non può essere ereditata da nessuna delle sue risorse secondarie.

Aggiornare le parti della documentazione

Per aggiornare le parti della documentazione di qualsiasi tipo di API entità, invia una PATCH richiesta su un'DocumentationPartistanza di un identificatore di parte specificato per sostituire la `properties` mappa esistente con una nuova.

```
PATCH /restapis/4wk1k4onj3/documentation/parts/part_id HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "RESOURCE_PATH",
    "value" : "NEW_properties_VALUE_AS_JSON_STRING"
  } ]
}
```

La risposta con esito positivo restituisce un codice di stato `200 OK` con un payload che include l'istanza `DocumentationPart` aggiornata nel payload.

Puoi aggiornare più parti della documentazione in una singola richiesta `PATCH`.

Elencare le parti della documentazione

Per elencare le parti della documentazione di qualsiasi tipo di API entità, invia una GET richiesta su una DocumentationPartsraccolta.

```
GET /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret
```

La risposta con esito positivo restituisce un codice di stato `200 OK` con un payload che include le istanze `DocumentationPart` disponibili nel payload.

Pubblica API la documentazione utilizzando il API Gateway REST API

Per pubblicare la documentazione di unaAPI, crea, aggiorna o ottieni un'istantanea della documentazione, quindi associa l'istantanea della documentazione a una fase. API Quando si crea un'istantanea della documentazione, è anche possibile associarla contemporaneamente a una API fase.

Argomenti

- Create un'istantanea della documentazione e associatela a una fase API
- Creare uno snapshot della documentazione
- Aggiornare uno snapshot della documentazione
- Ottenere uno snapshot della documentazione
- Associate un'istantanea della documentazione a una fase API
- Scaricare uno snapshot della documentazione associato a una fase

Create un'istantanea della documentazione e associatela a una fase API

Per creare un'istantanea delle parti API della documentazione di una pagina e associarla contemporaneamente a una API fase, invia la seguente `POST` richiesta:

```
POST /restapis/restapi_id/documentation/versions HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "documentationVersion" : "1.0.0",
    "stageName": "prod",
    "description" : "My API Documentation v1.0.0"
}
```

Se con esito positivo, l'operazione restituisce una risposta `200 OK` contenente la nuova istanza `DocumentationVersion` creata come payload.

In alternativa, puoi creare un'istantanea della documentazione senza associarla prima a una API fase e poi chiamare restapi:update per associare l'istantanea a una fase specificata. API Puoi inoltre aggiornare o interrogare uno snapshot della documentazione esistente e quindi aggiornare la relativa associazione della fase. Le fasi vengono illustrate nelle prossime quattro sezioni.

Creare uno snapshot della documentazione

Per creare un'istantanea delle parti della documentazione API di an, create una nuova DocumentationVersionrisorsa e aggiungetela alla raccolta di: DocumentationVersionsAPI

```
POST /restapis/restapi_id/documentation/versions HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "documentationVersion" : "1.0.0",
```

```
    "description" : "My API Documentation v1.0.0"
}
```

Se con esito positivo, l'operazione restituisce una risposta `200 OK` contenente la nuova istanza `DocumentationVersion` creata come payload.

Aggiornare uno snapshot della documentazione

È possibile aggiornare un'istantanea della documentazione solo modificando la `description` proprietà della risorsa corrispondente. DocumentationVersion L'esempio seguente mostra come aggiornare la descrizione dello snapshot della documentazione come identificata dal relativo identificatore di versione, *`version`*, ad esempio `1.0.0`.

```
PATCH /restapis/restapi_id/documentation/versions/version HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "patchOperations": [{
        "op": "replace",
        "path": "/description",
        "value": "My API for testing purposes."
    }]
}
```

Se con esito positivo, l'operazione restituisce una risposta `200 OK` contenente l'istanza `DocumentationVersion` aggiornata come payload.

Ottenere uno snapshot della documentazione

Per ottenere un'istantanea della documentazione, invia una `GET` richiesta relativa alla risorsa specificata. DocumentationVersion L'esempio seguente mostra come ottenere uno snapshot della documentazione di un dato identificatore di versione, 1.0.0.

```
GET /restapis/<restapi_id>/documentation/versions/1.0.0 HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
```

```
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret
```

Associate un'istantanea della documentazione a una fase API

Per pubblicare la API documentazione, associate un'istantanea della documentazione a una API fase. È necessario aver già creato una API fase prima di associare la versione della documentazione alla fase.

Per associare un'istantanea della documentazione a uno API stage utilizzando il APIGateway REST API, chiamate l'operazione stage:update per impostare la versione della documentazione desiderata sulla proprietà: `stage.documentationVersion`

```
PATCH /restapis/RESTAPI_ID/stages/STAGE_NAME
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "patchOperations": [{
        "op": "replace",
        "path": "/documentationVersion",
        "value": "VERSION_IDENTIFIER"
    }]
}
```

Scaricare uno snapshot della documentazione associato a una fase

Dopo aver associato una versione delle parti della documentazione a uno stage, è possibile esportare le parti della documentazione insieme alle definizioni delle API entità, in un file esterno, utilizzando la console API Gateway, il API Gateway RESTAPI, uno dei suoi SDKs o for Gateway. AWS CLI API Il processo è lo stesso utilizzato per l'APIesportazione di. Il formato del file esportato può essere JSON o. YAML

Utilizzando il API Gateway RESTAPI, potete anche impostare in modo esplicito il parametro di `extension=documentation,integrations,authorizers` query per includere le parti della API documentazione, API le integrazioni e gli autorizzatori in un'esportazione. API Per impostazione

predefinita, le parti della documentazione sono incluse, ma le integrazioni e gli autorizzatori sono esclusi quando si esporta un. API L'output predefinito di un'APIesportazione è adatto per la distribuzione della documentazione.

Per esportare la API documentazione in un API file JSON Open esterno utilizzando il API Gateway RESTAPI, inviate la seguente `GET` richiesta:

```
GET /restapis/restapi_id/stages/stage_name/exports/swagger?extensions=documentation
 HTTP/1.1
Accept: application/json
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret
```

Qui, l'`x-amazon-apigateway-documentation`oggetto contiene le parti della documentazione e le definizioni delle API entità contengono le proprietà della documentazione supportate da OpenAPI. L'output non include dettagli sull'integrazione o sulle autorizzazioni Lambda (note in precedenza come autorizzazioni ad hoc). Per includere entrambi i dettagli, imposta `extensions=integrations,authorizers,documentation`. Per includere i dettagli delle integrazioni ma non delle autorizzazioni, imposta `extensions=integrations,documentation`.

È necessario impostare l'`Accept:application/json`intestazione nella richiesta per visualizzare il risultato in un JSON file. Per produrre l'YAMLoutput, modifica l'intestazione della richiesta in. `Accept:application/yaml`

Ad esempio, vedremo un API che espone un `GET` metodo semplice sulla risorsa root ()`/`. Questo API ha quattro API entità definite in un file a API definizione aperta, una per ciascuno dei `RESPONSE` tipi `APIMODEL`,`METHOD`, e. Una parte della documentazione è stata aggiunta a ciascuna delle entità `API`, `METHOD` e `RESPONSE`. Richiamando il precedente comando di esportazione della documentazione, otteniamo il seguente risultato, con le parti della documentazione elencate all'interno dell'`x-amazon-apigateway-documentation`oggetto come estensione di un file Open standard. API

OpenAPI 3.0

```
{
   "openapi": "3.0.0",
   "info": {
```

```
      "description": "API info description",
      "version": "2016-11-22T22:39:14Z",
      "title": "doc",
      "x-bar": "API info x-bar"
   },
   "paths": {
      "/": {
         "get": {
            "description": "Method description.",
            "responses": {
               "200": {
                  "description": "200 response",
                  "content": {
                     "application/json": {
                        "schema": {
                           "$ref": "#/components/schemas/Empty"
                        }
                     }
                  }
               }
            },
            "x-example": "x- Method example"
         },
         "x-bar": "resource x-bar"
      }
   },
   "x-amazon-apigateway-documentation": {
      "version": "1.0.0",
      "createdDate": "2016-11-22T22:41:40Z",
      "documentationParts": [
         {
            "location": {
               "type": "API"
            },
            "properties": {
               "description": "API description",
               "foo": "API foo",
               "x-bar": "API x-bar",
               "info": {
                  "description": "API info description",
                  "version": "API info version",
                  "foo": "API info foo",
                  "x-bar": "API info x-bar"
               }
```

```
            }
         },
         {
            "location": {
               "type": "METHOD",
               "method": "GET"
            },
            "properties": {
               "description": "Method description.",
               "x-example": "x- Method example",
               "foo": "Method foo",
               "info": {
                  "version": "method info version",
                  "description": "method info description",
                  "foo": "method info foo"
               }
            }
         },
         {
            "location": {
               "type": "RESOURCE"
            },
            "properties": {
               "description": "resource description",
               "foo": "resource foo",
               "x-bar": "resource x-bar",
               "info": {
                  "description": "resource info description",
                  "version": "resource info version",
                  "foo": "resource info foo",
                  "x-bar": "resource info x-bar"
               }
            }
         }
      ]
   },
   "x-bar": "API x-bar",
   "servers": [
      {
         "url": "https://rznaap68yi.execute-api.ap-southeast-1.amazonaws.com/
{basePath}",
         "variables": {
            "basePath": {
              "default": "/test"
```

```
            }
         }
      }
   ],
   "components": {
      "schemas": {
         "Empty": {
            "type": "object",
            "title": "Empty Schema"
         }
      }
   }
}
```

OpenAPI 2.0

```
{
  "swagger" : "2.0",
  "info" : {
    "description" : "API info description",
    "version" : "2016-11-22T22:39:14Z",
    "title" : "doc",
    "x-bar" : "API info x-bar"
  },
  "host" : "rznaap68yi.execute-api.ap-southeast-1.amazonaws.com",
  "basePath" : "/test",
  "schemes" : [ "https" ],
  "paths" : {
    "/" : {
      "get" : {
        "description" : "Method description.",
        "produces" : [ "application/json" ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "schema" : {
              "$ref" : "#/definitions/Empty"
            }
          }
        },
        "x-example" : "x- Method example"
      },
      "x-bar" : "resource x-bar"
```

```
      }
    },
    "definitions" : {
      "Empty" : {
        "type" : "object",
        "title" : "Empty Schema"
      }
    },
    "x-amazon-apigateway-documentation" : {
      "version" : "1.0.0",
      "createdDate" : "2016-11-22T22:41:40Z",
      "documentationParts" : [ {
        "location" : {
          "type" : "API"
        },
        "properties" : {
          "description" : "API description",
          "foo" : "API foo",
          "x-bar" : "API x-bar",
          "info" : {
            "description" : "API info description",
            "version" : "API info version",
            "foo" : "API info foo",
            "x-bar" : "API info x-bar"
          }
        }
      }, {
        "location" : {
          "type" : "METHOD",
          "method" : "GET"
        },
        "properties" : {
          "description" : "Method description.",
          "x-example" : "x- Method example",
          "foo" : "Method foo",
          "info" : {
            "version" : "method info version",
            "description" : "method info description",
            "foo" : "method info foo"
          }
        }
      }, {
        "location" : {
          "type" : "RESOURCE"
```

```
      },
      "properties" : {
        "description" : "resource description",
        "foo" : "resource foo",
        "x-bar" : "resource x-bar",
        "info" : {
          "description" : "resource info description",
          "version" : "resource info version",
          "foo" : "resource info foo",
          "x-bar" : "resource info x-bar"
        }
      }
    } ]
  },
  "x-bar" : "API x-bar"
}
```

Per un attributo Open API -compliant definito nella `properties` mappa di una parte della documentazione, API Gateway inserisce l'attributo nella definizione di entità associata. API Un attributo di `x-`*`something`* è un'estensione Open standard. API Questa estensione viene propagata nella definizione dell'APIentità. Ad esempio, consulta l'attributo `x-example` per il metodo `GET`. Un attributo like non `foo` fa parte della API specifica Open e non viene inserito nelle definizioni di API entità associate.

Se uno strumento per il rendering della documentazione (ad esempio, Open API UI) analizza le definizioni delle API entità per estrarre gli attributi della documentazione, gli eventuali attributi non API conformi a Open di un'istanza `DocumentationPart` 'non sono `properties` disponibili per lo strumento. Tuttavia, se uno strumento di rendering della documentazione analizza l'`x-amazon-apigateway-documentation`oggetto per ottenere contenuti o se lo strumento chiama restapi:documentation-parts e documentenationpart:by-id per recuperare parti della documentazione da API Gateway, tutti gli attributi della documentazione sono disponibili per la visualizzazione dello strumento.

Per esportare la documentazione con le definizioni delle entità contenenti i dettagli di integrazione in un file Open, inviate la seguente richiesta: API JSON API `GET`

```
GET /restapis/restapi_id/stages/stage_name/exports/swagger?
extensions=integrations,documentation HTTP/1.1
Accept: application/json
```

```
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret
```

Per esportare la documentazione con le definizioni delle API entità contenenti i dettagli delle integrazioni e degli autorizzatori in un API file YAML Open, invia la seguente `GET` richiesta:

```
GET /restapis/restapi_id/stages/stage_name/exports/swagger?
extensions=integrations,authorizers,documentation HTTP/1.1
Accept: application/yaml
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret
```

Per utilizzare la console API Gateway per esportare e scaricare la documentazione pubblicata di anAPI, segui le istruzioni riportate in. Esportazione REST API tramite la console API Gateway

Importa API la documentazione

Analogamente all'importazione delle definizioni di API entità, è possibile importare parti della documentazione da un API file Open esterno in un API Gateway API in. Le parti della to-be-imported documentazione all'interno dell'x-amazon-apigateway-documentation oggettoestensione vengono specificate in un file di API definizione aperto valido. L'importazione della documentazione non modifica le definizioni di API entità esistenti.

È possibile unire le parti di documentazione appena specificate in parti di documentazione esistenti in API Gateway o sovrascrivere le parti della documentazione esistenti. In questa `MERGE` modalità, una nuova parte della documentazione definita nel API file Open viene aggiunta alla `DocumentationParts` raccolta di. API Se un elemento `DocumentationPart` importato esiste già, un attributo importato sostituisce quello esistente se i due sono diversi. Altri attributi della documentazione esistenti rimangono inalterati. In questa `OVERWRITE` modalità, l'intera `DocumentationParts` raccolta viene sostituita in base al file di API definizione aperto importato.

Importazione di parti della documentazione utilizzando il API Gateway REST API

Per importare API la documentazione utilizzando il API Gateway RESTAPI, chiamate l'operazione documentationpart:import. L'esempio seguente mostra come sovrascrivere le parti della documentazione esistenti di un API con un unico `GET /` metodo, restituendo una risposta in caso di successo. `200 OK`

OpenAPI 3.0

```
PUT /restapis/<restapi_id>/documentation/parts&mode=overwrite&failonwarnings=true
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
   "openapi": "3.0.0",
   "info": {
      "description": "description",
      "version": "1",
      "title": "doc"
   },
   "paths": {
      "/": {
         "get": {
            "description": "Method description.",
            "responses": {
               "200": {
                  "description": "200 response",
                  "content": {
                     "application/json": {
                        "schema": {
                           "$ref": "#/components/schemas/Empty"
                        }
                     }
                  }
               }
            }
         }
      }
   },
```

```
  "x-amazon-apigateway-documentation": {
     "version": "1.0.3",
     "documentationParts": [
        {
           "location": {
              "type": "API"
           },
           "properties": {
              "description": "API description",
              "info": {
                 "description": "API info description 4",
                 "version": "API info version 3"
              }
           }
        },
        {
           "location": {
              "type": "METHOD",
              "method": "GET"
           },
           "properties": {
              "description": "Method description."
           }
        },
        {
           "location": {
              "type": "MODEL",
              "name": "Empty"
           },
           "properties": {
              "title": "Empty Schema"
           }
        },
        {
           "location": {
              "type": "RESPONSE",
              "method": "GET",
              "statusCode": "200"
           },
           "properties": {
              "description": "200 response"
           }
        }
     ]
```

```
   },
   "servers": [
      {
         "url": "/"
      }
   ],
   "components": {
      "schemas": {
         "Empty": {
            "type": "object",
            "title": "Empty Schema"
         }
      }
   }
}
```

OpenAPI 2.0

```
PUT /restapis/<restapi_id>/documentation/parts&mode=overwrite&failonwarnings=true
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
  "swagger": "2.0",
  "info": {
    "description": "description",
    "version": "1",
    "title": "doc"
  },
  "host": "",
  "basePath": "/",
  "schemes": [
    "https"
  ],
  "paths": {
    "/": {
      "get": {
        "description": "Method description.",
        "produces": [
```

```
          "application/json"
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
        }
      }
    }
  },
  "definitions": {
    "Empty": {
      "type": "object",
      "title": "Empty Schema"
    }
  },
  "x-amazon-apigateway-documentation": {
    "version": "1.0.3",
    "documentationParts": [
      {
        "location": {
          "type": "API"
        },
        "properties": {
          "description": "API description",
          "info": {
            "description": "API info description 4",
            "version": "API info version 3"
          }
        }
      },
      {
        "location": {
          "type": "METHOD",
          "method": "GET"
        },
        "properties": {
          "description": "Method description."
        }
      },
      {
```

```
        "location": {
          "type": "MODEL",
          "name": "Empty"
        },
        "properties": {
          "title": "Empty Schema"
        }
      },
      {
        "location": {
          "type": "RESPONSE",
          "method": "GET",
          "statusCode": "200"
        },
        "properties": {
          "description": "200 response"
        }
      }
    ]
  }
}
```

In caso di esito positivo, questa richiesta restituisce una risposta 200 OK contenente l'elemento `DocumentationPartId` importato nel payload.

```
{
  "ids": [
    "kg3mth",
    "796rtf",
    "zhek4p",
    "5ukm9s"
  ]
}
```

Inoltre, potete anche chiamare restapi:import o restapi:put, fornendo le parti della documentazione nell'`x-amazon-apigateway-documentation`oggetto come parte del file Open di input della definizione. API API Per escludere le parti della documentazione dall'APIimportazione, impostate i parametri della query di richiesta. `ignore=documentation`

Importazione di parti della documentazione utilizzando la console API Gateway

Le seguenti istruzioni descrivono come importare le parti della documentazione.

Per utilizzare la console per importare parti di documentazione API da un file esterno

1. Nel riquadro di navigazione principale scegli Documentazione.
2. Seleziona Importa.
3. Se è disponibile la documentazione esistente, seleziona Sovrascrivi o Unisci per la nuova documentazione.
4. Seleziona Scegli il file per caricare un file da un'unità oppure immettere il contenuto di un file nella vista del file. Per un esempio, consulta il payload della richiesta di esempio in Importazione di parti della documentazione utilizzando il API Gateway REST API.
5. Scegli in che modo gestire gli avvisi durante l'importazione. Seleziona Avvisi di errore o Ignora avvisi. Per ulteriori informazioni, consulta the section called “Errori e avvisi derivanti dall'importazione in Gateway API API”.
6. Seleziona Importa.

Controlla l'accesso alla API documentazione in API Gateway

Se disponi di un team dedicato alla documentazione per scrivere e modificare la API documentazione, puoi configurare autorizzazioni di accesso separate per gli sviluppatori (per API lo sviluppo) e per i tuoi autori o redattori (per lo sviluppo dei contenuti). Questo è particolarmente appropriato quando un fornitore di terze parti è coinvolto nella creazione della documentazione.

Per concedere al team addetto alla documentazione l'accesso per creare, aggiornare e pubblicare la API documentazione, puoi assegnare un IAM ruolo al team addetto alla documentazione secondo la seguente IAM politica, dove *account_id* è l'ID dell' AWS account del team addetto alla documentazione.

```
{
  "Version": "2012-10-17",
  "Statement": [

    {
      "Sid": "StmtDocPartsAddEditViewDelete",
      "Effect": "Allow",
      "Action": [
        "apigateway:GET",
        "apigateway:PUT",
        "apigateway:POST",
        "apigateway:PATCH",
```

```
        "apigateway:DELETE"
      ],
      "Resource": [
        "arn:aws:apigateway::account_id:/restapis/*/documentation/*"
      ]
    }
  ]
}
```

Per informazioni sull'impostazione delle autorizzazioni per accedere alle risorse del API Gateway, consultathe section called “Come funziona Amazon API Gateway con IAM”.

Genera SDKs per REST APIs in API Gateway

Per chiamarti REST API in un modo specifico per piattaforma o lingua, devi generare la piattaforma o la lingua SDK specifica di. API Il tuo viene generato SDK dopo averlo creato, testato e distribuito su uno stage. API Attualmente, API Gateway supporta la generazione di an SDK for an API in Java JavaScript, Java per Android, Objective-C o Swift per iOS e Ruby.

Questa sezione spiega come generare un SDK Gateway. API API Dimostra inoltre come utilizzare il generato SDK in un'app Java, un'app Java per Android, Objective-C e Swift per iOS e un'app. JavaScript

Per facilitare la discussione, utilizziamo questo API Gateway API, che espone questa funzione Lambda di Simple Calculator.

Prima di procedere, crea o importa il file API e distribuiscilo almeno una volta in Gateway. API Per istruzioni, consulta Implementazione REST APIs in Gateway API.

Argomenti

- Funzione Lambda del calcolatore semplice
- API calcolatore semplice in API Gateway
- Definizione OpenAPI dell'API del calcolatore semplice
- Genera il codice Java SDK di un API in Gateway API
- Genera l'Android di un in SDK Gateway API API
- Genera l'iOS SDK di un API in API Gateway
- Genera il file di un in JavaScript SDK Gateway REST API API
- Genera il Ruby SDK di un in Gateway API API

- Genera SDKs per e API utilizza AWS CLI i comandi in Gateway API

Funzione Lambda del calcolatore semplice

A scopi illustrativi, verrà utilizzata una funzione Lambda in Node.js che esegue le operazioni binarie di addizione, sottrazione, moltiplicazione e divisione.

Argomenti

- Formato di input della funzione Lambda del calcolatore semplice
- Formato di output della funzione Lambda del calcolatore semplice
- Implementazione della funzione Lambda del calcolatore semplice

Formato di input della funzione Lambda del calcolatore semplice

Questa funzione riceve un input nel formato seguente:

```
{ "a": "Number", "b": "Number", "op": "string"}
```

dove op può essere un valore `(+, -, *, /, add, sub, mul, div)` qualsiasi.

Formato di output della funzione Lambda del calcolatore semplice

Quando un'operazione ha esito positivo, restituisce il risultato nel formato seguente:

```
{ "a": "Number", "b": "Number", "op": "string", "c": "Number"}
```

dove `c` contiene il risultato del calcolo.

Implementazione della funzione Lambda del calcolatore semplice

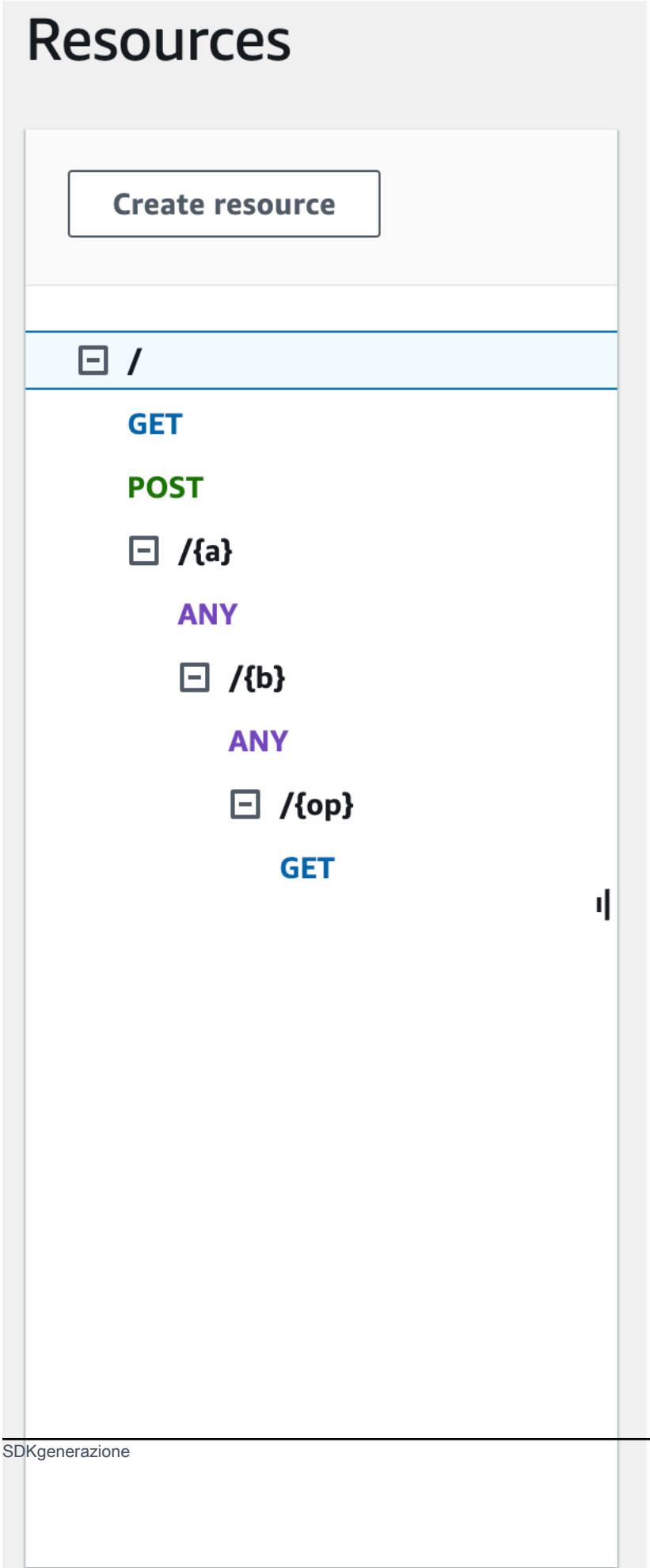
L'implementazione della funzione Lambda è la seguente:

```
export const handler = async function (event, context) {
  console.log("Received event:", JSON.stringify(event));

  if (
    event.a === undefined ||
    event.b === undefined ||
    event.op === undefined
  ) {
```

```
    return "400 Invalid Input";
  }

  const res = {};
  res.a = Number(event.a);
  res.b = Number(event.b);
  res.op = event.op;
  if (isNaN(event.a) || isNaN(event.b)) {
    return "400 Invalid Operand";
  }
  switch (event.op) {
    case "+":
    case "add":
      res.c = res.a + res.b;
      break;
    case "-":
    case "sub":
      res.c = res.a - res.b;
      break;
    case "*":
    case "mul":
      res.c = res.a * res.b;
      break;
    case "/":
    case "div":
      if (res.b == 0) {
        return "400 Divide by Zero";
      } else {
        res.c = res.a / res.b;
      }
      break;
    default:
      return "400 Invalid Operator";
  }

  return res;
};
```

API calcolatore semplice in API Gateway

L'API calcolatore semplice espone tre metodi (GET, POST, GET) per richiamare la the section called “Funzione Lambda del calcolatore semplice”. Di seguito è illustrata una rappresentazione grafica di quest'API:


Resources
Create resource
/
GET
POST
/{a}
ANY
/{b}
ANY
/{op}
GET

Questi tre metodi mostrano modi diversi per fornire l'input per la funzione Lambda di back-end per eseguire la stessa operazione:

- Il metodo `GET /?a=...&b=...&op=...` usa i parametri di query per specificare l'input.
- Il metodo `POST /` usa un payload JSON `{"a":"Number", "b":"Number", "op":"string"}` per specificare l'input.
- Il metodo `GET /{a}/{b}/{op}` usa i parametri di percorso per specificare l'input.

Se non è definito, API Gateway genera il nome del metodo SDK corrispondente combinando le parti relative al metodo HTTP e al percorso. La parte del percorso root (`/`) è detta `Api Root`. Ad esempio, il nome del metodo SDK Java predefinito per il metodo API `GET /?a=...&b=...&op=...` è `getAB0p`, il nome del metodo SDK predefinito per `POST /` è `postApiRoot` e il nome del metodo SDK predefinito per `GET /{a}/{b}/{op}` è `getAB0p`. La convenzione può essere personalizzata per i singoli SDK. Consulta la documentazione dell'origine dell'SDK generato per informazioni sui nomi di metodi SDK specifici.

È possibile e consigliabile sostituire i nomi dei metodi SDK predefiniti specificando la proprietà operationName in ogni metodo API. Questa operazione viene eseguita durante la creazione del metodo API o l'aggiornamento del metodo API con l'API REST di API Gateway. Nella definizione Swagger dell'API è possibile impostare `operationId` per ottenere lo stesso risultato.

Prima di illustrare come invocare questi metodi usando un SDK generato da API Gateway per questa API, esaminiamo brevemente come eseguire la configurazione. Per istruzioni dettagliate, consulta Sviluppa REST APIs in API Gateway. Se non hai mai usato API Gateway, consulta prima Scegli un tutorial di integrazione AWS Lambda.

Creazione di modelli per l'input e l'output

Per specificare un input fortemente tipizzato nell'SDK, creiamo un modello `Input` per l'API. Per descrivere il tipo di dati del corpo della risposta, creiamo un modello `Output` e un modello `Result`.

Per creare i modelli per l'input, l'output e il risultato

1. Nel riquadro di navigazione principale seleziona Modelli.
2. Scegli Crea modello.
3. Per Nome, immetti **`input`**.
4. Per Tipo di contenuto inserisci **`application/json`**.

Se non viene trovato alcun tipo di contenuto corrispondente, la convalida della richiesta non viene eseguita. Per utilizzare lo stesso modello indipendentemente dal tipo di contenuti, inserisci **`$default`**.

5. Per Schema modello immetti il seguente modello:

```
{
    "$schema" : "$schema": "http://json-schema.org/draft-04/schema#",
    "type":"object",
    "properties":{
        "a":{"type":"number"},
        "b":{"type":"number"},
        "op":{"type":"string"}
    },
    "title":"Input"
}
```

6. Scegli Crea modello.
7. Ripeti le seguenti fasi per creare un modello `Output` e un modello `Result`.

   Per il modello `Output` immetti in Schema modello quanto segue:

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "properties": {
        "c": {"type":"number"}
    },
    "title": "Output"
}
```

   Per il modello `Result` immetti in Schema modello quanto segue. Sostituisci l'ID API `abc123` con il tuo ID API.

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type":"object",
    "properties":{
        "input":{
            "$ref":"https://apigateway.amazonaws.com/restapis/abc123/models/Input"
        },
```

```
        "output":{
            "$ref":"https://apigateway.amazonaws.com/restapis/abc123/models/Output"
        }
    },
    "title":"Result"
}
```

Configurazione dei parametri di query del metodo GET /

Per il metodo `GET /?a=..&b=..&op=..`, i parametri di query sono dichiarati in Method Request (Richiesta metodo):

Per configurare i parametri della stringa di query GET/URL

1. Nella sezione Richiesta metodo scegli Modifica per il metodo `GET` sulla risorsa root (`/`).
2. Scegli Parametri della stringa di query URL ed effettua le seguenti operazioni:
   a. Scegliere Add query string (Aggiungi stringa di query).
   b. Per Nome, immetti **a**.
   c. Mantieni Obbligatorio e Caching disattivati.
   d. Mantieni disattivata l'opzione Caching.

   Ripeti le stesse fasi e crea una stringa di query denominata **b** e una stringa di query denominata **op**.
3. Selezionare Salva.

Configurazione del modello di dati per il payload come input nel back-end

Per il metodo `POST /`, creiamo il modello `Input` e lo aggiungiamo alla richiesta del metodo per definire la forma dei dati di input.

Per configurare il modello di dati per il payload come input nel back-end

1. Nella sezione Richiesta metodo scegli Modifica per il metodo `POST` sulla risorsa root (`/`).
2. Scegli Corpo della richiesta.
3. Scegliere Add model (Aggiungi modello).
4. Per Tipo di contenuto inserisci **`application/json`**.

5. Per Modello seleziona Input.
6. Selezionare Salva.

Con questo modello, i clienti dell'API possono chiamare l'SDK per specificare l'input creando un'istanza di un oggetto `Input`. Senza questo modello, i clienti dovrebbero creare un oggetto dizionario per rappresentare l'input JSON per la funzione Lambda.

Configurazione del modello di dati per l'output del risultato dal back-end

Per tutti e tre i metodi, creiamo il modello `Result` e lo aggiungiamo in `Method Response` per il metodo per definire la forma dell'output restituito dalla funzione Lambda.

Per configurare il modello di dati per l'output del risultato dal back-end

1. Seleziona la risorsa /{a}/{b}/{op}, quindi scegli il metodo GET.
2. Nella scheda Risposta metodo scegli Modifica in Risposta 200.
3. In Corpo della risposta scegli Aggiungi modello.
4. Per Tipo di contenuto inserisci **`application/json`**.
5. Per Modello seleziona Risultato.
6. Selezionare Salva.

Con questo modello, i clienti dell'API possono analizzare un output con esito positivo leggendo le proprietà di un oggetto `Result`. Senza questo modello, i clienti dovrebbero creare un oggetto dizionario per rappresentare l'output JSON.

Definizione OpenAPI dell'API del calcolatore semplice

Di seguito è riportata una definizione OpenAPI dell'API del calcolatore semplice. Puoi importarla nel tuo account. Tuttavia, dopo l'importazione, dovrai reimpostare le autorizzazioni basate su risorse per la funzione Lambda. A questo scopo, riseleziona la funzione Lambda creata nel tuo account da Integration Request (Richiesta di integrazione) nella console API Gateway. La console API Gateway reimposterà le autorizzazioni richieste. In alternativa, puoi utilizzare AWS Command Line Interface per il comando Lambda add-permission.

OpenAPI 2.0

```
{
  "swagger": "2.0",
```

```
  "info": {
    "version": "2016-09-29T20:27:30Z",
    "title": "SimpleCalc"
  },
  "host": "t6dve4zn25.execute-api.us-west-2.amazonaws.com",
  "basePath": "/demo",
  "schemes": [
    "https"
  ],
  "paths": {
    "/": {
      "get": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "op",
            "in": "query",
            "required": false,
            "type": "string"
          },
          {
            "name": "a",
            "in": "query",
            "required": false,
            "type": "string"
          },
          {
            "name": "b",
            "in": "query",
            "required": false,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Result"
            }
```

```
          }
        },
        "x-amazon-apigateway-integration": {
          "requestTemplates": {
            "application/json": "#set($inputRoot = $input.path('$'))\n{\n
  \"a\" : $input.params('a'),\n  \"b\" : $input.params('b'),\n  \"op\" :
 \"$input.params('op')\"\n}"
          },
          "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:Calc/invocations",
          "passthroughBehavior": "when_no_templates",
          "httpMethod": "POST",
          "responses": {
            "default": {
              "statusCode": "200",
              "responseTemplates": {
                "application/json": "#set($inputRoot = $input.path('$'))\n{\n
 \"input\" : {\n    \"a\" : $inputRoot.a,\n    \"b\" : $inputRoot.b,\n    \"op\" :
 \"$inputRoot.op\"\n  },\n  \"output\" : {\n    \"c\" : $inputRoot.c\n  }\n}"
              }
            }
          },
          "type": "aws"
        }
      },
      "post": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "in": "body",
            "name": "Input",
            "required": true,
            "schema": {
              "$ref": "#/definitions/Input"
            }
          }
        ],
        "responses": {
          "200": {
```

```
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Result"
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:Calc/invocations",
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST",
          "responses": {
            "default": {
              "statusCode": "200",
              "responseTemplates": {
                "application/json": "#set($inputRoot = $input.path('$'))\n{\n
 \"input\" : {\n    \"a\" : $inputRoot.a,\n    \"b\" : $inputRoot.b,\n    \"op\" :
 \"$inputRoot.op\"\n  },\n  \"output\" : {\n    \"c\" : $inputRoot.c\n  }\n}"
              }
            }
          },
          "type": "aws"
        }
      }
    },
    "/{a}": {
      "x-amazon-apigateway-any-method": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "a",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "404": {
            "description": "404 response"
```

```
          }
        },
        "x-amazon-apigateway-integration": {
          "requestTemplates": {
            "application/json": "{\"statusCode\": 200}"
          },
          "passthroughBehavior": "when_no_match",
          "responses": {
            "default": {
              "statusCode": "404",
              "responseTemplates": {
                "application/json": "{ \"Message\" : \"Can't $context.httpMethod
 $context.resourcePath\" }"
              }
            }
          },
          "type": "mock"
        }
      }
    },
    "/{a}/{b}": {
      "x-amazon-apigateway-any-method": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "a",
            "in": "path",
            "required": true,
            "type": "string"
          },
          {
            "name": "b",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "404": {
```

```
            "description": "404 response"
          }
        },
        "x-amazon-apigateway-integration": {
          "requestTemplates": {
            "application/json": "{\"statusCode\": 200}"
          },
          "passthroughBehavior": "when_no_match",
          "responses": {
            "default": {
              "statusCode": "404",
              "responseTemplates": {
                "application/json": "{ \"Message\" : \"Can't $context.httpMethod
$context.resourcePath\" }"
              }
            }
          },
          "type": "mock"
        }
      }
    },
    "/{a}/{b}/{op}": {
      "get": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "a",
            "in": "path",
            "required": true,
            "type": "string"
          },
          {
            "name": "b",
            "in": "path",
            "required": true,
            "type": "string"
          },
          {
            "name": "op",
```

```
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Result"
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "requestTemplates": {
            "application/json": "#set($inputRoot = $input.path('$'))\n{\n
  \"a\" : $input.params('a'),\n  \"b\" : $input.params('b'),\n  \"op\" :
 \"$input.params('op')\"\n}"
          },
          "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:Calc/invocations",
          "passthroughBehavior": "when_no_templates",
          "httpMethod": "POST",
          "responses": {
            "default": {
              "statusCode": "200",
              "responseTemplates": {
                "application/json": "#set($inputRoot = $input.path('$'))\n{\n
 \"input\" : {\n    \"a\" : $inputRoot.a,\n    \"b\" : $inputRoot.b,\n    \"op\" :
 \"$inputRoot.op\"\n  },\n  \"output\" : {\n    \"c\" : $inputRoot.c\n  }\n}"
              }
            }
          },
          "type": "aws"
        }
      }
    }
  },
  "definitions": {
    "Input": {
      "type": "object",
      "properties": {
        "a": {
          "type": "number"
```

```
        },
        "b": {
          "type": "number"
        },
        "op": {
          "type": "string"
        }
      },
      "title": "Input"
    },
    "Output": {
      "type": "object",
      "properties": {
        "c": {
          "type": "number"
        }
      },
      "title": "Output"
    },
    "Result": {
      "type": "object",
      "properties": {
        "input": {
          "$ref": "#/definitions/Input"
        },
        "output": {
          "$ref": "#/definitions/Output"
        }
      },
      "title": "Result"
    }
  }
}
```

OpenAPI 3.0

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "SimpleCalc",
    "version" : "2016-09-29T20:27:30Z"
  },
  "servers" : [ {
```

```
    "url" : "https://t6dve4zn25.execute-api.us-west-2.amazonaws.com/{basePath}",
    "variables" : {
      "basePath" : {
        "default" : "demo"
      }
    }
  } ],
  "paths" : {
    "/{a}/{b}" : {
      "x-amazon-apigateway-any-method" : {
        "parameters" : [ {
          "name" : "a",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "b",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "404" : {
            "description" : "404 response",
            "content" : { }
          }
        },
        "x-amazon-apigateway-integration" : {
          "type" : "mock",
          "responses" : {
            "default" : {
              "statusCode" : "404",
              "responseTemplates" : {
                "application/json" : "{ \"Message\" : \"Can't $context.httpMethod
$context.resourcePath\" }"
              }
            }
          },
          "requestTemplates" : {
            "application/json" : "{\"statusCode\": 200}"
```

```
          },
          "passthroughBehavior" : "when_no_match"
        }
      }
    },
    "/{a}/{b}/{op}" : {
      "get" : {
        "parameters" : [ {
          "name" : "a",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "b",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "op",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "content" : {
              "application/json" : {
                "schema" : {
                  "$ref" : "#/components/schemas/Result"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : {
          "type" : "aws",
          "httpMethod" : "POST",
```

```
          "uri" : "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:111122223333:function:Calc/invocations",
          "responses" : {
            "default" : {
              "statusCode" : "200",
              "responseTemplates" : {
                "application/json" : "#set($inputRoot = $input.path('$'))\n{\n
 \"input\" : {\n    \"a\" : $inputRoot.a,\n    \"b\" : $inputRoot.b,\n    \"op\" :
 \"$inputRoot.op\"\n  },\n  \"output\" : {\n    \"c\" : $inputRoot.c\n  }\n}"
              }
            }
          },
          "requestTemplates" : {
            "application/json" : "#set($inputRoot = $input.path('$'))\n{\n
  \"a\" : $input.params('a'),\n  \"b\" : $input.params('b'),\n  \"op\" :
 \"$input.params('op')\"\n}"
          },
          "passthroughBehavior" : "when_no_templates"
        }
      }
    },
    "/" : {
      "get" : {
        "parameters" : [ {
          "name" : "op",
          "in" : "query",
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "a",
          "in" : "query",
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "b",
          "in" : "query",
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "200" : {
```

```
            "description" : "200 response",
            "content" : {
              "application/json" : {
                "schema" : {
                  "$ref" : "#/components/schemas/Result"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : {
          "type" : "aws",
          "httpMethod" : "POST",
          "uri" : "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:111122223333:function:Calc/invocations",
          "responses" : {
            "default" : {
              "statusCode" : "200",
              "responseTemplates" : {
                "application/json" : "#set($inputRoot = $input.path('$'))\n{\n
 \"input\" : {\n    \"a\" : $inputRoot.a,\n    \"b\" : $inputRoot.b,\n    \"op\" :
 \"$inputRoot.op\"\n  },\n  \"output\" : {\n    \"c\" : $inputRoot.c\n  }\n}"
              }
            }
          },
          "requestTemplates" : {
            "application/json" : "#set($inputRoot = $input.path('$'))\n{\n
  \"a\" : $input.params('a'),\n  \"b\" : $input.params('b'),\n  \"op\" :
 \"$input.params('op')\"\n}"
          },
          "passthroughBehavior" : "when_no_templates"
        }
      },
      "post" : {
        "requestBody" : {
          "content" : {
            "application/json" : {
              "schema" : {
                "$ref" : "#/components/schemas/Input"
              }
            }
          },
          "required" : true
        },
```

```
        "responses" : {
          "200" : {
            "description" : "200 response",
            "content" : {
              "application/json" : {
                "schema" : {
                  "$ref" : "#/components/schemas/Result"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : {
          "type" : "aws",
          "httpMethod" : "POST",
          "uri" : "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:111122223333:function:Calc/invocations",
          "responses" : {
            "default" : {
              "statusCode" : "200",
              "responseTemplates" : {
                "application/json" : "#set($inputRoot = $input.path('$'))\n{\n
 \"input\" : {\n    \"a\" : $inputRoot.a,\n    \"b\" : $inputRoot.b,\n    \"op\" :
 \"$inputRoot.op\"\n  },\n  \"output\" : {\n    \"c\" : $inputRoot.c\n  }\n}"
              }
            }
          },
          "passthroughBehavior" : "when_no_match"
        }
      }
    },
    "/{a}" : {
      "x-amazon-apigateway-any-method" : {
        "parameters" : [ {
          "name" : "a",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "404" : {
            "description" : "404 response",
```

```
            "content" : { }
          }
        },
        "x-amazon-apigateway-integration" : {
          "type" : "mock",
          "responses" : {
            "default" : {
              "statusCode" : "404",
              "responseTemplates" : {
                "application/json" : "{ \"Message\" : \"Can't $context.httpMethod
 $context.resourcePath\" }"
              }
            }
          },
          "requestTemplates" : {
            "application/json" : "{\"statusCode\": 200}"
          },
          "passthroughBehavior" : "when_no_match"
        }
      }
    }
  },
  "components" : {
    "schemas" : {
      "Input" : {
        "title" : "Input",
        "type" : "object",
        "properties" : {
          "a" : {
            "type" : "number"
          },
          "b" : {
            "type" : "number"
          },
          "op" : {
            "type" : "string"
          }
        }
      },
      "Output" : {
        "title" : "Output",
        "type" : "object",
        "properties" : {
          "c" : {
```

```
            "type" : "number"
          }
        }
      },
      "Result" : {
        "title" : "Result",
        "type" : "object",
        "properties" : {
          "input" : {
            "$ref" : "#/components/schemas/Input"
          },
          "output" : {
            "$ref" : "#/components/schemas/Output"
          }
        }
      }
    }
  }
}
```

Genera il codice Java SDK di un API in Gateway API

La procedura seguente mostra come generare il codice Java SDK di un API in API Gateway.

Per generare il codice Java SDK di un API in API Gateway

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli un RESTAPI.
3. Scegliere Stages (Fasi).
4. Nel riquadro Fasi, seleziona il nome della fase.
5. Apri il menu delle azioni dello stage, quindi scegli Genera SDK.
6. In Piattaforma, scegli la piattaforma Java ed effettua le seguenti operazioni:

   a. Per Service Name, specifica il nome del tuoSDK. Ad esempio, **`SimpleCalcSdk`**. Questo diventa il nome della tua classe SDK client. Il nome corrisponde al `<name>` tag contenuto `<project>` nel file pom.xml, che si trova nella cartella SDK del progetto. Non includere i trattini.

b. Per Java Package Name, specifica un nome di pacchetto per il tuoSDK. Ad esempio, **`examples.aws.apig.simpleCalc.sdk`**. Questo nome di pacchetto viene utilizzato come spazio dei nomi della libreriaSDK. Non includere i trattini.

c. In Java Build System (Sistema di compilazione Java) immettere **`maven`** o **`gradle`** per specificare il sistema di compilazione.

d. Per Java Group Id, inserisci un identificatore di gruppo per il tuo progetto. SDK Ad esempio, specifica **`my-apig-api-examples`**. Questo identificatore corrisponde al `<groupId>` tag contenuto `<project>` nel `pom.xml` file, che si trova nella cartella SDK del progetto del file.

e. Per Java Artifact Id, inserisci un identificatore di artefatto per il tuo progetto. SDK Ad esempio, specifica **`simple-calc-sdk`**. Questo identificatore corrisponde al `<artifactId>` tag contenuto nel `pom.xml` file, che si trova `<project>` nella cartella del progetto del file. SDK

f. In Java Artifact Version (Versione artefatto Java), immettere la stringa identificatore della versione. Ad esempio, **`1.0.0`**. Questo identificatore di versione corrisponde al `<version>` tag contenuto `<project>` nel `pom.xml` file, che si trova nella cartella SDK del progetto del file.

g. In Source Code License Text (Testo della licenza del codice sorgente), immettere il testo della licenza del codice sorgente, se disponibile.

7. Scegli Genera SDK, quindi segui le istruzioni sullo schermo per scaricare il file SDK generato da API Gateway.

Segui le istruzioni Utilizzo di un SDK Java generato da API Gateway per un'API REST per utilizzare il file generatoSDK.

Ogni volta che aggiorni unAPI, devi ridistribuirlo API e SDK rigenerarlo per includerlo.

Genera l'Android di un in SDK Gateway API API

La procedura seguente mostra come generare l'Android SDK di un API in API Gateway.

Per generare l'Android SDK di un API in API Gateway

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli un RESTAPI.
3. Scegliere Stages (Fasi).
4. Nel riquadro Fasi, seleziona il nome della fase.

5. Apri il menu delle azioni dello stage, quindi scegli Genera SDK.
6. In Piattaforma, scegli la piattaforma Android ed effettua le seguenti operazioni:
   a. In Group ID (ID gruppo), immettere l'identificatore univoco per il progetto corrispondente che viene usato nel file `pom.xml`, ad esempio **`com.mycompany`**.
   b. In Invoker package (Pacchetto di invoker), immettere lo spazio dei nomi per le classi client generate, ad esempio **`com.mycompany.clientsdk`**.
   c. In Artifact ID (ID artefatto), immettere il nome del file .jar compilato senza la versione che viene usato nel file `pom.xml`, ad esempio **`aws-apigateway-api-sdk`**.
   d. In Artifact version (Versione artefatto), immettere il numero di versione dell'artefatto per il cliente generato che Viene utilizzato nel `pom.xml` file e dovrebbe seguire un *`major.minor.patch`* modello (ad esempio,**`1.0.0`**).
7. Scegli Genera SDK, quindi segui le istruzioni sullo schermo per scaricare il file SDK generato da API Gateway.

Segui le istruzioni Utilizzo di un SDK Android generato da API Gateway per un'API REST per utilizzare il file generatoSDK.

Ogni volta che aggiorni unAPI, devi ridistribuirlo API e SDK rigenerarlo per includerlo.

Genera l'iOS SDK di un API in API Gateway

La procedura seguente mostra come generare l'iOS SDK di un API in API Gateway.

Per generare l'iOS SDK di un API in API Gateway

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli un RESTAPI.
3. Scegliere Stages (Fasi).
4. Nel riquadro Fasi, seleziona il nome della fase.
5. Apri il menu delle azioni dello stage, quindi scegli Genera SDK.
6. In Piattaforma, scegli la piattaforma iOS (Objective-C) o iOS (Swift) ed effettua le seguenti operazioni:
   - Digitare un prefisso univoco nella casella Prefix (Prefisso).

L'effetto del prefisso è il seguente: se lo assegnate, ad esempio, **`SIMPLE_CALC`** come prefisso per i `result` modelli SimpleCalcAPIwith`input`, and`output`, il valore generato SDK conterrà la `SIMPLE_CALCSimpleCalcClient` classe che incapsula, incluse le richieste/risposte del metodoAPI. SDK Inoltre, il valore generato SDK conterrà le `SIMPLE_CALCresult` classi `SIMPLE_CALCinputSIMPLE_CALCoutput`, e che rappresenteranno rispettivamente l'input, l'output e i risultati, per rappresentare l'input della richiesta e l'output della risposta. Per ulteriori informazioni, consulta Uso dell'SDK iOS generato da API Gateway per un'API REST in Objective-C o Swift.

7. Scegli Genera SDK, quindi segui le istruzioni sullo schermo per scaricare il file SDK generato da API Gateway.

Segui le istruzioni Uso dell'SDK iOS generato da API Gateway per un'API REST in Objective-C o Swift per utilizzare il file generatoSDK.

Ogni volta che aggiorni unAPI, devi ridistribuirlo API e SDK rigenerarlo per includerlo.

Genera il file di un in JavaScript SDK Gateway REST API API

La procedura seguente mostra come generare un API in API Gateway. JaveScript SDK

Per generare il JavaScript SDK di un API Gateway API in

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli un RESTAPI.
3. Scegliere Stages (Fasi).
4. Nel riquadro Fasi, seleziona il nome della fase.
5. Apri il menu delle azioni dello stage, quindi scegli Genera SDK.
6. Per Piattaforma, scegliete la JavaScriptpiattaforma.
7. Scegli Genera SDK, quindi segui le istruzioni sullo schermo per scaricare il file SDK generato da API Gateway.

Segui le istruzioni Usa un file JavaScript SDK generato da API Gateway per un REST API per utilizzare il file generatoSDK.

Ogni volta che aggiorni unAPI, devi ridistribuirlo API e SDK rigenerarlo per includerlo.

Genera il Ruby SDK di un in Gateway API API

La procedura seguente mostra come generare il Ruby SDK di un API in API Gateway.

Per generare il Ruby SDK di un API in Gateway API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli un RESTAPI.
3. Scegliere Stages (Fasi).
4. Nel riquadro Fasi, seleziona il nome della fase.
5. Apri il menu delle azioni dello stage, quindi scegli Genera SDK.
6. In Piattaforma, scegli la piattaforma Ruby ed effettua le seguenti operazioni:
   a. Per Service Name, specifica il nome del tuoSDK. Ad esempio, **`SimpleCalc`**. Viene utilizzato per generare lo spazio dei nomi Ruby Gem del tuo. API Deve essere composto solo da lettere, (`a-zA-Z`), senza caratteri speciali o numeri.
   b. Per Ruby Gem Name, specifica il nome di Ruby Gem per contenere il codice sorgente generato per il tuo. SDK API Per impostazione predefinita, è il nome del servizio in minuscolo più il suffisso `-sdk`, ad esempio **`simplecalc-sdk`**.
   c. In Ruby Gem Version (Versione Ruby Gem) specificare il numero di versione del Ruby Gem generato. Per impostazione predefinita, è impostato su `1.0.0`.
7. Scegli Genera SDK, quindi segui le istruzioni sullo schermo per scaricare il file generato da Gateway. SDK API

Segui le istruzioni Usa un Ruby SDK generato da API Gateway per un REST API per utilizzare il file generatoSDK.

Ogni volta che aggiorni unAPI, devi ridistribuirlo API e SDK rigenerarlo per includerlo.

Genera SDKs per e API utilizza AWS CLI i comandi in Gateway API

È possibile utilizzare AWS CLI per generare e scaricare un file SDK di un file API per una piattaforma supportata chiamando il comando get-sdk. Di seguito è riportata la dimostrazione per alcune delle piattaforme supportate.

Argomenti

- Generate e scaricate Java per Android utilizzando SDK AWS CLI

- Genera e scarica il file JavaScript SDK utilizzando il AWS CLI
- Genera e scarica Ruby SDK usando il AWS CLI

Generate e scaricate Java per Android utilizzando SDK AWS CLI

Per generare e scaricare un file Java per Android SDK generato da API Gateway of an API (`udpuvvzbkc`) in una determinata fase (`test`), chiamate il comando come segue:

```
aws apigateway get-sdk \
            --rest-api-id udpuvvzbkc \
            --stage-name test \
            --sdk-type android \
            --parameters groupId='com.mycompany',\
                invokerPackage='com.mycompany.myApiSdk',\
                artifactId='myApiSdk',\
                artifactVersion='0.0.1' \
            ~/apps/myApi/myApi-android-sdk.zip
```

L'ultimo input di `~/apps/myApi/myApi-android-sdk.zip` è il percorso del SDK file scaricato denominato`myApi-android-sdk.zip`.

Genera e scarica il file JavaScript SDK utilizzando il AWS CLI

Per generare e scaricare un file JavaScript SDK generato da API Gateway di an API (`udpuvvzbkc`) in una determinata fase (`test`), chiamate il comando come segue:

```
aws apigateway get-sdk \
            --rest-api-id udpuvvzbkc \
            --stage-name test \
            --sdk-type javascript \
            ~/apps/myApi/myApi-js-sdk.zip
```

L'ultimo input di `~/apps/myApi/myApi-js-sdk.zip` è il percorso del SDK file scaricato denominato`myApi-js-sdk.zip`.

Genera e scarica Ruby SDK usando il AWS CLI

Per generare e scaricare un Ruby SDK of an API (`udpuvvzbkc`) in una determinata fase (`test`), chiamate il comando come segue:

```
aws apigateway get-sdk \
```

```
            --rest-api-id udpuvvzbkc \
            --stage-name test  \
            --sdk-type ruby \
            --parameters service.name=myApiRubySdk,ruby.gem-name=myApi,ruby.gem-
version=0.01 \
            ~/apps/myApi/myApi-ruby-sdk.zip
```

L'ultimo input di `~/apps/myApi/myApi-ruby-sdk.zip` è il percorso del SDK file scaricato denominato`myApi-ruby-sdk.zip`.

Successivamente, mostriamo come utilizzare il generato SDK per chiamare il sottostanteAPI. Per ulteriori informazioni, consulta Richiama le API REST in API Gateway.

Vendi le tue API API Gateway tramite Marketplace AWS

Dopo aver creato, testato e distribuito le API, puoi impacchettarle in un piano di utilizzo API Gateway e vendere il piano come prodotto Software as a Service (SaaS) tramite. Marketplace AWS Gli acquirenti di API che si abbonano alla tua offerta di prodotti vengono fatturati in Marketplace AWS base al numero di richieste effettuate al piano di utilizzo.

Per vendere le tue API Marketplace AWS, devi configurare il canale di vendita per l'integrazione Marketplace AWS con API Gateway. In generale, ciò comporta l'inserimento del prodotto su Marketplace AWS, l'impostazione di un ruolo IAM con politiche appropriate per consentire ad API Gateway di inviare metriche di utilizzo a Marketplace AWS, l'associazione di un Marketplace AWS prodotto a un piano di utilizzo API Gateway e l'associazione di un Marketplace AWS acquirente a una chiave API Gateway. I dettagli vengono approfonditi nelle sezioni seguenti.

Per ulteriori informazioni sulla vendita della tua API come prodotto SaaS Marketplace AWS, consulta la Guida per l'Marketplace AWS utente.

Argomenti

- Inizializzazione dell'integrazione di Marketplace AWS con API Gateway
- Gestione della sottoscrizione del cliente ai piani di utilizzo

Inizializzazione dell'integrazione di Marketplace AWS con API Gateway

Le seguenti attività riguardano l'inizializzazione unica dell' Marketplace AWS integrazione con API Gateway, che consente di vendere le API come prodotto SaaS.

Pubblica un prodotto su Marketplace AWS

Per inserire il tuo piano di utilizzo come prodotto SaaS, invia un modulo di caricamento prodotto tramite Marketplace AWS. Il prodotto deve contenere una dimensione denominata `apigateway` del tipo `requests`. Questa dimensione definisce price-per-request e viene utilizzata da API Gateway per misurare le richieste alle API.

Creazione del ruolo di misurazione

Crea un ruolo IAM denominato `ApiGatewayMarketplaceMeteringRole` con le seguenti policy di esecuzione e policy di attendibilità. Questo ruolo consente ad API Gateway di inviare metriche di utilizzo Marketplace AWS a tuo nome.

Policy di esecuzione del ruolo di misurazione

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "aws-marketplace:BatchMeterUsage",
        "aws-marketplace:ResolveCustomer"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

Policy di relazione di fiducia del ruolo di misurazione

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "apigateway.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
```

```
}
```

Associa il piano di utilizzo al prodotto Marketplace AWS

Quando offri un prodotto su Marketplace AWS, ricevi un codice Marketplace AWS prodotto. Per integrare API Gateway con Marketplace AWS, associa il tuo piano di utilizzo al codice Marketplace AWS del prodotto. Puoi abilitare l'associazione impostando il `productCode`campo `UsagePlan` dell'API Gateway sul codice del Marketplace AWS prodotto, utilizzando la console API Gateway, l'API REST API Gateway, l'API AWS CLI for API Gateway o l' AWS SDK per API Gateway. Nell'esempio di codice seguente viene utilizzata l'API REST dell'API Gateway:

```
PATCH /usageplans/USAGE_PLAN_ID
Host: apigateway.region.amazonaws.com
Authorization: ...

{
    "patchOperations" : [{
        "path" : "/productCode",
        "value" : "MARKETPLACE_PRODUCT_CODE",
        "op" : "replace"
    }]
}
```

Gestione della sottoscrizione del cliente ai piani di utilizzo

Le seguenti attività sono gestite dall'applicazione del portale sviluppatore.

Quando un cliente si abbona al tuo prodotto tramite Marketplace AWS, Marketplace AWS inoltra una `POST` richiesta all'URL degli abbonamenti SaaS che hai registrato al momento dell'offerta del prodotto. Marketplace AWS La richiesta `POST` presenta un parametro `x-amzn-marketplace-token` contenente le informazioni sull'acquirente. Segui le istruzioni riportate in Onboarding dei clienti SaaS per gestire questo reindirizzamento nell'applicazione del portale per gli sviluppatori.

In risposta alla richiesta di iscrizione di un cliente, Marketplace AWS invia una `subscribe-success` notifica a un argomento di Amazon SNS a cui puoi abbonarti. (Consulta Onboarding dei clienti SaaS). Per accettare la richiesta di abbonamento del cliente, gestisci la `subscribe-success` notifica creando o recuperando una chiave API Gateway per il cliente, associando la chiave API Marketplace AWS-provisioned `customerId` del cliente alle chiavi API e quindi aggiungendo la chiave API al tuo piano di utilizzo.

Quando la richiesta di sottoscrizione del cliente è completa, l'applicazione del portale per gli sviluppatori deve mostrare al cliente la chiave API associata e informarlo che la chiave API deve essere inclusa nell'intestazione `x-api-key` quando si effettuano le richieste alle API.

Quando un cliente annulla un abbonamento a un piano di utilizzo, Marketplace AWS invia una `unsubscribe-success` notifica all'argomento SNS. Per portare a termine il processo di cancellazione di un cliente, devi gestire la notifica `unsubscribe-success` eliminando le chiavi API del cliente dal piano di utilizzo.

Per autorizzare un cliente ad accedere a un piano di utilizzo

Per autorizzare un cliente ad accedere al tuo piano di utilizzo, usa l'API di API Gateway per recuperare o creare una chiave API per il cliente e aggiungerla al piano di utilizzo.

L'esempio seguente mostra come chiamare l'API REST di API Gateway per creare una nuova chiave API con un Marketplace AWS `customerId` valore specifico (*`MARKETPLACE_CUSTOMER_ID`*).

```
POST apikeys HTTP/1.1
Host: apigateway.region.amazonaws.com
Authorization: ...

{
  "name" : "my_api_key",
  "description" : "My API key",
  "enabled" : "false",
  "stageKeys" : [ {
    "restApiId" : "uycll6xg9a",
    "stageName" : "prod"
  } ],
  "customerId" : "MARKETPLACE_CUSTOMER_ID"
}
```

L'esempio seguente mostra come ottenere una chiave API con un valore specifico (MARKETPLACE_CUSTOMER_ID). Marketplace AWS customerId

```
GET apikeys?customerId=MARKETPLACE_CUSTOMER_ID HTTP/1.1
Host: apigateway.region.amazonaws.com
Authorization: ...
```

Per aggiungere una chiave API a un piano di utilizzo, crea un `UsagePlanKey` con la chiave API per il relativo piano di utilizzo. Il seguente esempio mostra come ottenere tale risultato utilizzando

l'API REST di API Gateway, dove `n371pt` è l'ID del piano di utilizzo e `q5ugs7qjjh` è un esempio di `keyId` API restituito dagli esempi precedenti.

```
POST /usageplans/n371pt/keys HTTP/1.1
Host: apigateway.region.amazonaws.com
Authorization: ...

{
    "keyId": "q5ugs7qjjh",
    "keyType": "API_KEY"
}
```

Per associare un cliente a una chiave API

È necessario aggiornare il `customerId` campo 's' con l'ID `ApiKey`cliente del Marketplace AWS cliente. In questo modo, la chiave API viene associata al cliente Marketplace AWS , attivando così misurazione e fatturazione per l'acquirente. Il seguente esempio di codice invoca l'API REST API Gateway per ottenere tale risultato.

```
PATCH /apikeys/q5ugs7qjjh
Host: apigateway.region.amazonaws.com
Authorization: ...

{
    "patchOperations" : [{
        "path" : "/customerId",
        "value" : "MARKETPLACE_CUSTOMER_ID",
        "op" : "replace"
    }]
}
```

Proteggi le tue API REST in API Gateway

API Gateway fornisce diversi modi per proteggere l'API da determinate minacce, ad esempio utenti malintenzionati o picchi di traffico. Puoi proteggere l'API utilizzando strategie come la generazione di certificati SSL, la configurazione di un firewall per applicazioni Web, l'impostazione di destinazioni di limitazione e l'accesso all'API solo da un cloud privato virtuale (VPC, Virtual Private Cloud). In questa sezione viene descritto come abilitare queste funzionalità utilizzando API Gateway.

Argomenti

- Come attivare l'TLSautenticazione reciproca per il tuo REST APIs in API Gateway
- Genera e configura un SSL certificato per l'autenticazione backend in API Gateway
- Utilizzalo AWS WAF per proteggere le tue API REST in API Gateway
- Limita le richieste alle tue API REST per una migliore velocità di trasmissione in API Gateway
- Privato REST APIs in API Gateway

Come attivare l'TLSautenticazione reciproca per il tuo REST APIs in API Gateway

TLSL'autenticazione reciproca richiede l'autenticazione bidirezionale tra il client e il server. Con MutualTLS, i client devono presentare certificati X.509 per verificare la propria identità e accedere al API Mutual TLS è un requisito comune per l'Internet of Things (IoT) e business-to-business le applicazioni.

È possibile utilizzare Mutual TLS insieme ad altre operazioni di autorizzazione e autenticazione supportate da API Gateway. APIGateway inoltra i certificati forniti dai client agli autorizzatori Lambda e alle integrazioni di backend.

> **Important**
>
> Per impostazione predefinita, i client possono richiamarti API utilizzando l'`execute-api`endpoint che Gateway genera per te. API API Per garantire che i clienti possano accedere al tuo API solo utilizzando un nome di dominio personalizzato con mutuoTLS, disabilita l'endpoint predefinito`execute-api`. Per ulteriori informazioni, consulta the section called “Disabilitazione dell'endpoint predefinito”.

Argomenti

- Prerequisiti per la mutua TLS
- Configurazione di Mutual TLS per un nome di dominio personalizzato
- Invoca un utente API utilizzando un nome di dominio personalizzato che richiede mutuo TLS
- Aggiornamento del truststore
- Disabilita mutuo TLS
- Risolvi i problemi relativi a Mutual per TLS REST API

Prerequisiti per la mutua TLS

Per configurare mutuo TLS è necessario:

- Un nome di dominio personalizzato regionale
- Almeno un certificato configurato AWS Certificate Manager per il tuo nome di dominio personalizzato
- Un truststore configurato e caricato in Amazon S3

Nomi di dominio personalizzati

Per abilitare TLS il mutuo per a RESTAPI, devi configurare un nome di dominio personalizzato per il tuoAPI. È possibile abilitare Mutual TLS per un nome di dominio personalizzato e quindi fornire il nome di dominio personalizzato ai client. Per accedere e utilizzare un nome API di dominio personalizzato con TLS attivazione reciproca, i client devono presentare certificati attendibili nelle API richieste. Puoi trovare ulteriori informazioni all'indirizzo the section called “Nomi di dominio personalizzati”.

Utilizzo dei certificati AWS Certificate Manager emessi

È possibile richiedere un certificato pubblicamente attendibile direttamente da ACM o importare certificati pubblici o autofirmati. Per configurare un certificato inACM, vai a ACM. Se desideri importare un certificato, continua a leggere la sezione seguente.

Utilizzo di un AWS Private Certificate Authority certificato o importato

Per utilizzare un certificato importato in ACM o da AWS Private Certificate Authority mutuoTLS, API Gateway necessita di un certificato `ownershipVerificationCertificate` emesso daACM. Questo certificato di proprietà viene utilizzato solo per verificare che disponi delle autorizzazioni per l'utilizzo del nome di dominio. Non viene utilizzato per la TLS stretta di mano. Se non ne hai già uno`ownershipVerificationCertificate`, vai https://console.aws.amazon.com/acm/a configurarne uno.

Dovrai mantenere la validità di questo certificato per tutta la durata del nome di dominio. Se un certificato scade e il rinnovo automatico non riesce, tutti gli aggiornamenti al nome di dominio verranno bloccati. Dovrai aggiornare il `ownershipVerificationCertificateArn` con un `ownershipVerificationCertificate` valido prima di poter apportare altre modifiche. `ownershipVerificationCertificate`Non può essere utilizzato come certificato server per un

altro TLS dominio comune in API Gateway. Se un certificato viene reimportato direttamente inACM, l'emittente deve rimanere lo stesso.

Configurazione del truststore

I truststore sono file di testo con un'estensione `.pem`. Sono elenchi attendibili di certificati provenienti da autorità di certificazione. Per utilizzare mutuoTLS, crea un truststore di certificati X.509 di cui ti fidi per accedere ai tuoi. API

Devi includere nel truststore l'intera catena di attendibilità, a partire dal certificato della CA emittente, fino al certificato emesso da una CA radice. APIGateway accetta certificati client emessi da qualsiasi CA presente nella catena di fiducia. I certificati possono provenire da autorità di certificazione pubbliche o private. I certificati possono avere una lunghezza massima della catena di quattro. È inoltre possibile fornire certificati autofirmati. I seguenti algoritmi sono supportati nel truststore:

- SHA-256 o superiore
- RSA-2048 o superiore
- ECDSA-256 o -384 ECDSA

APIGateway convalida una serie di proprietà del certificato. È possibile utilizzare gli autorizzatori Lambda per eseguire controlli aggiuntivi quando un client richiama un certificatoAPI, incluso verificare se un certificato è stato revocato. APIGateway convalida le seguenti proprietà:

Validation	Descrizione
Sintassi X.509	Il certificato deve soddisfare i requisiti di sintassi X.509.
Integrità	Il contenuto del certificato non deve essere stato modificato rispetto a quello firmato dall'autorità di certificazione del truststore.
Validity	Il periodo di validità del certificato deve essere attuale.
Concatenamento di nomi/concatenamento di chiavi	I nomi e gli oggetti dei certificati devono formare una catena ininterrotta. I certificati

Validation	Descrizione
	possono avere una lunghezza massima della catena di quattro.

Caricare il truststore in un bucket Amazon S3 in un singolo file

Di seguito è riportato un esempio di come potrebbe presentarsi un file .pem.

Example certificates.pem

```
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
...
```

Il AWS CLI comando seguente viene caricato nel tuo `certificates.pem` bucket Amazon S3.

```
aws s3 cp certificates.pem s3://bucket-name
```

Il tuo bucket Amazon S3 deve avere l'autorizzazione di lettura per API Gateway per consentire API a Gateway di accedere al tuo truststore.

Configurazione di Mutual TLS per un nome di dominio personalizzato

Per configurare mutuo TLS per un RESTAPI, è necessario utilizzare un nome di dominio personalizzato regionale per il proprioAPI, con una politica `TLS_1_2` di sicurezza. Per ulteriori informazioni sulla scelta di una politica di sicurezza, consultathe section called “Scegli una politica di sicurezza”.

> **Note**
>
> Mutual TLS non è supportato per uso privatoAPIs.

Dopo aver caricato il tuo truststore su Amazon S3, puoi configurare il tuo nome di dominio personalizzato per utilizzare mutuo. TLS Incolla quanto segue (barre incluse) in un terminale:

```
aws apigateway create-domain-name --region us-east-2 \
    --domain-name api.example.com \
    --regional-certificate-arn arn:aws:acm:us-
east-2:123456789012:certificate/123456789012-1234-1234-1234-12345678 \
    --endpoint-configuration types=REGIONAL \
    --security-policy TLS_1_2 \
    --mutual-tls-authentication truststoreUri=s3://bucket-name/key-name
```

Dopo aver creato il nome di dominio, devi configurare i DNS record e le mappature dei percorsi di base per le operazioni. API Per ulteriori informazioni, consulta Configura un nome di dominio personalizzato regionale in API Gateway.

Invoca un utente API utilizzando un nome di dominio personalizzato che richiede mutuo TLS

Per richiamare un account API con mutuo TLS abilitato, i client devono presentare un certificato attendibile nella API richiesta. Quando un client tenta di richiamare il tuoAPI, API Gateway cerca l'emittente del certificato client nel tuo truststore. APIAffinché Gateway possa procedere con la richiesta, l'emittente del certificato e l'intera catena di fiducia fino al certificato CA principale devono trovarsi nel tuo truststore.

Il seguente comando `curl` di esempio seguente invia una richiesta `api.example.com,` che include `my-cert.pem` nella richiesta. `my-key.key` è la chiave privata del certificato.

```
curl -v --key ./my-key.key --cert ./my-cert.pem api.example.com
```

Il tuo API viene invocato solo se il truststore considera attendibile il certificato. Le seguenti condizioni faranno sì che API Gateway fallisca l'TLShandshake e neghi la richiesta con un codice di stato. `403` Se il tuo certificato:

- non è attendibile
- è scaduto
- non utilizza un algoritmo supportato

> **Note**
>
> APIGateway non verifica se un certificato è stato revocato.

Aggiornamento del truststore

Per aggiornare i certificati nel truststore, carica un nuovo bundle di certificati in Amazon S3. Potrai quindi aggiornare il nome di dominio personalizzato per utilizzare il certificato aggiornato.

Usa il Controllo delle versioni di Amazon S3 per mantenere più versioni del truststore. Quando aggiorni il tuo nome di dominio personalizzato per utilizzare una nuova versione del truststore, API Gateway restituisce avvisi se i certificati non sono validi.

APIGateway emette avvisi sui certificati solo quando aggiorni il nome di dominio. APIGateway non ti avvisa se un certificato precedentemente caricato scade.

Il AWS CLI comando seguente aggiorna un nome di dominio personalizzato per utilizzare una nuova versione di truststore.

```
aws apigateway update-domain-name \
    --domain-name api.example.com \
    --patch-operations op='replace',path='/mutualTlsAuthentication/
truststoreVersion',value='abcdef123'
```

Disabilita mutuo TLS

Per disabilitare TLS il mutuo per un nome di dominio personalizzato, rimuovi il truststore dal tuo nome di dominio personalizzato, come mostrato nel comando seguente.

```
aws apigateway update-domain-name \
    --domain-name api.example.com \
    --patch-operations op='replace',path='/mutualTlsAuthentication/
truststoreUri',value=''
```

Risolvi i problemi relativi a Mutual per TLS REST API

Di seguito vengono forniti consigli per la risoluzione di errori e problemi che potrebbero verificarsi quando si attiva Mutual. TLS

Risoluzione dei problemi relativi agli avvisi dei certificati

Quando si crea un nome di dominio personalizzato con mutuoTLS, API Gateway restituisce avvisi se i certificati nel truststore non sono validi. Questo può verificarsi anche quando aggiorni un nome di dominio personalizzato all'utilizzo di un nuovo truststore. Gli avvisi indicano il problema con il certificato e l'oggetto del certificato che ha generato l'avviso. Mutual TLS è ancora abilitato per teAPI, ma alcuni client potrebbero non essere in grado di accedervi. API

Per identificare il certificato che ha generato l'avviso, devi decodificare i certificati nel truststore. Puoi utilizzare strumenti come `openssl` per decodificare i certificati e identificarne gli oggetti.

Il comando seguente visualizza il contenuto di un certificato, incluso l'oggetto:

```
openssl x509 -in certificate.crt -text -noout
```

Aggiorna o rimuovi i certificati che hanno prodotto gli avvisi, quindi carica un nuovo truststore su Amazon S3. Dopo aver caricato il nuovo truststore, aggiorna il nome di dominio personalizzato per utilizzarlo.

Risoluzione dei problemi relativi ai conflitti dei nomi di dominio

L'errore `"The certificate subject <certSubject> conflicts with an existing certificate from a different issuer."` indica che più autorità di certificazione hanno emesso un certificato per questo dominio. Per ogni soggetto del certificato, può esserci un solo emittente in API Gateway per i TLS domini reciproci. Dovrai ottenere tutti i certificati per tale oggetto tramite un unico emittente. Se il problema riguarda un certificato di cui non hai il controllo ma puoi provare la proprietà del nome di dominio, contatta AWS Support per aprire un ticket.

Risoluzione dei problemi relativi ai messaggi di stato dei nomi di dominio

`PENDING_CERTIFICATE_REIMPORT`: Ciò significa che hai reimportato un certificato ACM e la convalida non è riuscita perché il nuovo certificato ha un SAN (nome alternativo del soggetto) che non è coperto dal `ownershipVerificationCertificate` o l'oggetto o SANs nel certificato non copre il nome di dominio. Potrebbe esserci una configurazione errata o è stato importato un certificato non valido. È necessario reimportare un certificato valido in. ACM Per ulteriori informazioni sulla convalida, consulta Convalida della proprietà del dominio.

`PENDING_OWNERSHIP_VERIFICATION`: Ciò significa che il certificato precedentemente verificato è scaduto e non ACM è stato possibile rinnovarlo automaticamente. Dovrai rinnovare il certificato o

richiederne uno nuovo. Ulteriori informazioni sul rinnovo dei certificati sono disponibili nella guida alla risoluzione dei problemi relativi al ACM rinnovo gestito dei certificati.

Risolvi i problemi relativi al certificato restituito errato

Quando si esegue la migrazione di un certificato dedicato da un nome di dominio completo (FQDN) a un nome di dominio cliente wildcard, API Gateway potrebbe restituire il certificato per il nome di dominio FQDN anziché il nome di dominio wildcard.

Il comando seguente mostra quale certificato viene restituito da Gateway: API

```
openssl s_client -connect hostname:port
```

Se il certificato risultante è per ilFQDN, contatta AWS Support per aprire un ticket.

Genera e configura un SSL certificato per l'autenticazione backend in API Gateway

È possibile utilizzare API Gateway per generare un SSL certificato e quindi utilizzare la relativa chiave pubblica nel backend per verificare che HTTP le richieste al sistema di backend provengano da Gateway. API Ciò consente al HTTP backend di controllare e accettare solo le richieste che provengono da Amazon API Gateway, anche se il backend è accessibile pubblicamente.

> **Note**
>
> Alcuni server di backend potrebbero non supportare l'autenticazione SSL client come fa API Gateway e potrebbero restituire un SSL errore di certificato. Per un elenco di server back-end non compatibili, consulta the section called “Note importanti”.

I SSL certificati generati da API Gateway sono autofirmati e solo la chiave pubblica di un certificato è visibile nella console API Gateway o tramite. APIs

Argomenti

- Genera un certificato client utilizzando la console API Gateway
- Configura e API usa SSL i certificati
- Test della chiamata per verificare la configurazione del certificato del client

- Configura un HTTPS server di backend per verificare il certificato del client
- Rotazione di un certificato client in scadenza
- APIAutorità di certificazione supportate dal gateway HTTP e integrazioni HTTP proxy in Gateway API

Genera un certificato client utilizzando la console API Gateway

1. Apri la console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway/.
2. Scegli un RESTAPI.
3. Nel riquadro di navigazione principale scegli Certificati client.
4. Nel riquadro Certificati client scegli Genera certificato client.
5. (Facoltativo) In Description (Descrizione), immettere una descrizione.
6. Scegli Genera certificato per generare il certificato. APIGateway genera un nuovo certificato e lo restituisceGUID, insieme alla chiave pubblica con PEM codifica.

Ora sei pronto per configurare e API utilizzare il certificato.

Configura e API usa SSL i certificati

Queste istruzioni presuppongono che la procedura in sia già stata completat Genera un certificato client utilizzando la console API Gateway.

1. Nella console API Gateway, crea o apri un certificato client API per il quale desideri utilizzare il certificato client. Assicurati che sia API stato distribuito in una fase.
2. Nel riquadro di navigazione principale scegli Fasi.
3. Nella sezione Dettagli fase scegli Modifica.
4. Per Certificato client seleziona un certificato.
5. Seleziona Salvataggio delle modifiche.

   Se API è stato distribuito in precedenza nella console API Gateway, sarà necessario ridistribuirlo affinché le modifiche abbiano effetto. Per ulteriori informazioni, consulta the section called “Crea distribuzione”.

Dopo aver selezionato API e salvato un certificato, API Gateway utilizza il certificato per tutte le chiamate alle HTTP integrazioni nel tuo. API

Test della chiamata per verificare la configurazione del certificato del client

1. Scegli un API metodo. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda Test.
2. Per Certificato client seleziona un certificato.
3. Scegli Test (Esegui test).

APIGateway presenta il SSL certificato scelto per il HTTP backend per autenticare il. API

Configura un HTTPS server di backend per verificare il certificato del client

Queste istruzioni presuppongono che Genera un certificato client utilizzando la console API Gateway sia già stata completata e che sia stata scaricata una copia del certificato client. È possibile scaricare un certificato client `clientcertificate:by-id`chiamando il API Gateway REST API o `get-client-certificate`il. AWS CLI

Prima di configurare un HTTPS server di backend per verificare il SSL certificato client di API Gateway, è necessario aver ottenuto la chiave privata PEM con codifica e un certificato lato server fornito da un'autorità di certificazione attendibile.

Se il nome di dominio del server è`myserver.mydomain.com`, il valore del certificato del server deve essere o. CNAME `myserver.mydomain.com` `*.mydomain.com`

Tra le autorità di certificazione supportate figurano Let's Encrypt o una delle the section called “Autorità di certificazione supportate per l'integrazione con proxy HTTP HTTP”.

Ad esempio, supponiamo che il file del certificato client sia `apig-cert.pem` e che la chiave privata del server e i file di certificato siano, rispettivamente, `server-key.pem` e `server-cert.pem`. Per un server Node.js nel back-end, è possibile configurare il server in modo simile al seguente:

```
var fs = require('fs');
var https = require('https');
var options = {
    key: fs.readFileSync('server-key.pem'),
    cert: fs.readFileSync('server-cert.pem'),
    ca: fs.readFileSync('apig-cert.pem'),
    requestCert: true,
    rejectUnauthorized: true
};
```

```
https.createServer(options, function (req, res) {
    res.writeHead(200);
    res.end("hello world\n");
}).listen(443);
```

Per un'app node-express, puoi utilizzare i client-certificate-authmoduli per autenticare le richieste dei client con certificati PEM -encoded.

Per altri HTTPS server, consulta la documentazione relativa al server.

Rotazione di un certificato client in scadenza

Il certificato client generato da API Gateway è valido per 365 giorni. È necessario ruotare il certificato prima della scadenza di un certificato client su uno API stage per evitare tempi di inattività per. API È possibile verificare la data di scadenza del certificato chiamando:by-id clientCertificate del Gateway API oppure eseguendo il comando AWS CLI e ispezionando la proprietà REST API restituita. get-client-certificateexpirationDate

Per ruotare un certificato client, procedere come indicato di seguito:

1. Genera un nuovo certificato client chiamando clientcertificate:generate del Gateway o il comando di. API REST API AWS CLI generate-client-certificate In questo tutorial si presuppone che l'ID del nuovo certificato client sia `ndiqef`.
2. Aggiorna il server di back-end per includere il nuovo certificato client. Non rimuovere il certificato client esistente.

   Alcuni server potrebbero richiedere un riavvio per completare l'aggiornamento. Per ulteriori informazioni, consulta la documentazione del server per vedere se è necessario riavviare il server durante l'aggiornamento.
3. Aggiorna lo API stage per utilizzare il nuovo certificato client chiamando stage:update del API Gateway RESTAPI, con il nuovo client certificate ID (): `ndiqef`

   ```
   PATCH /restapis/{restapi-id}/stages/stage1 HTTP/1.1
   Content-Type: application/json
   Host: apigateway.us-east-1.amazonaws.com
   X-Amz-Date: 20170603T200400Z
   Authorization: AWS4-HMAC-SHA256 Credential=...

   {
   ```

```
  "patchOperations" : [
    {
        "op" : "replace",
        "path" : "/clientCertificateId",
        "value" : "ndiqef"
    }
  ]
}
```

o chiamando il CLI comando di update-stage.

4. Aggiorna il server di back-end per eliminare il vecchio certificato.
5. Elimina il vecchio certificato da API Gateway chiamando il clientcertificate:delete del API Gateway RESTAPI, specificando il () del clientCertificateId vecchio certificato: `a1b2c3`

```
DELETE /clientcertificates/a1b2c3
```

o chiamando il comando di: CLI delete-client-certificate

```
aws apigateway delete-client-certificate --client-certificate-id a1b2c3
```

Per ruotare un certificato client nella console per uno distribuito in precedenzaAPI, procedi come segue:

1. Nel riquadro di navigazione principale scegli Certificati client.
2. Nel riquadro Certificati client scegli Genera certificato.
3. Apri il certificato client API per cui desideri utilizzare il certificato client.
4. Scegli Fasi sotto quella selezionataAPI, quindi scegli una fase.
5. Nella sezione Dettagli fase scegli Modifica.
6. Per Certificato client seleziona il nuovo certificato.
7. Per salvare le impostazioni, scegli Salva modifiche.

   È necessario ridistribuire il API affinché le modifiche abbiano effetto. Per ulteriori informazioni, consulta the section called “Crea distribuzione”.

APIAutorità di certificazione supportate dal gateway HTTP e integrazioni HTTP proxy in Gateway API

L'elenco seguente mostra le autorità di certificazione supportate da API Gateway per le integrazioniHTTP, HTTP proxy e private.

```
Alias name: accvraiz1
  SHA1: 93:05:7A:88:15:C6:4F:CE:88:2F:FA:91:16:52:28:78:BC:53:64:17
  SHA256:
 9A:6E:C0:12:E1:A7:DA:9D:BE:34:19:4D:47:8A:D7:C0:DB:18:22:FB:07:1D:F1:29:81:49:6E:D1:04:38:41:1
Alias name: acraizfnmtrcm
  SHA1: EC:50:35:07:B2:15:C4:95:62:19:E2:A8:9A:5B:42:99:2C:4C:2C:20
  SHA256:
 EB:C5:57:0C:29:01:8C:4D:67:B1:AA:12:7B:AF:12:F7:03:B4:61:1E:BC:17:B7:DA:B5:57:38:94:17:9B:93:F
Alias name: actalis
  SHA1: F3:73:B3:87:06:5A:28:84:8A:F2:F3:4A:CE:19:2B:DD:C7:8E:9C:AC
  SHA256:
 55:92:60:84:EC:96:3A:64:B9:6E:2A:BE:01:CE:0B:A8:6A:64:FB:FE:BC:C7:AA:B5:AF:C1:55:B3:7F:D7:60:6
Alias name: actalisauthenticationrootca
  SHA1: F3:73:B3:87:06:5A:28:84:8A:F2:F3:4A:CE:19:2B:DD:C7:8E:9C:AC
  SHA256:
 55:92:60:84:EC:96:3A:64:B9:6E:2A:BE:01:CE:0B:A8:6A:64:FB:FE:BC:C7:AA:B5:AF:C1:55:B3:7F:D7:60:6
Alias name: addtrustclass1ca
  SHA1: CC:AB:0E:A0:4C:23:01:D6:69:7B:DD:37:9F:CD:12:EB:24:E3:94:9D
  SHA256:
 8C:72:09:27:9A:C0:4E:27:5E:16:D0:7F:D3:B7:75:E8:01:54:B5:96:80:46:E3:1F:52:DD:25:76:63:24:E9:A
Alias name: addtrustexternalca
  SHA1: 02:FA:F3:E2:91:43:54:68:60:78:57:69:4D:F5:E4:5B:68:85:18:68
  SHA256:
 68:7F:A4:51:38:22:78:FF:F0:C8:B1:1F:8D:43:D5:76:67:1C:6E:B2:BC:EA:B4:13:FB:83:D9:65:D0:6D:2F:F
Alias name: addtrustqualifiedca
  SHA1: 4D:23:78:EC:91:95:39:B5:00:7F:75:8F:03:3B:21:1E:C5:4D:8B:CF
  SHA256:
 80:95:21:08:05:DB:4B:BC:35:5E:44:28:D8:FD:6E:C2:CD:E3:AB:5F:B9:7A:99:42:98:8E:B8:F4:DC:D0:60:1
Alias name: affirmtrustcommercial
  SHA1: F9:B5:B6:32:45:5F:9C:BE:EC:57:5F:80:DC:E9:6E:2C:C7:B2:78:B7
  SHA256:
 03:76:AB:1D:54:C5:F9:80:3C:E4:B2:E2:01:A0:EE:7E:EF:7B:57:B6:36:E8:A9:3C:9B:8D:48:60:C9:6F:5F:A
Alias name: affirmtrustcommercialca
  SHA1: F9:B5:B6:32:45:5F:9C:BE:EC:57:5F:80:DC:E9:6E:2C:C7:B2:78:B7
  SHA256:
 03:76:AB:1D:54:C5:F9:80:3C:E4:B2:E2:01:A0:EE:7E:EF:7B:57:B6:36:E8:A9:3C:9B:8D:48:60:C9:6F:5F:A
Alias name: affirmtrustnetworking
```

```
  SHA1: 29:36:21:02:8B:20:ED:02:F5:66:C5:32:D1:D6:ED:90:9F:45:00:2F
  SHA256:
 0A:81:EC:5A:92:97:77:F1:45:90:4A:F3:8D:5D:50:9F:66:B5:E2:C5:8F:CD:B5:31:05:8B:0E:17:F3:F0:B4:1
Alias name: affirmtrustnetworkingca
  SHA1: 29:36:21:02:8B:20:ED:02:F5:66:C5:32:D1:D6:ED:90:9F:45:00:2F
  SHA256:
 0A:81:EC:5A:92:97:77:F1:45:90:4A:F3:8D:5D:50:9F:66:B5:E2:C5:8F:CD:B5:31:05:8B:0E:17:F3:F0:B4:1
Alias name: affirmtrustpremium
  SHA1: D8:A6:33:2C:E0:03:6F:B1:85:F6:63:4F:7D:6A:06:65:26:32:28:27
  SHA256:
 70:A7:3F:7F:37:6B:60:07:42:48:90:45:34:B1:14:82:D5:BF:0E:69:8E:CC:49:8D:F5:25:77:EB:F2:E9:3B:9
Alias name: affirmtrustpremiumca
  SHA1: D8:A6:33:2C:E0:03:6F:B1:85:F6:63:4F:7D:6A:06:65:26:32:28:27
  SHA256:
 70:A7:3F:7F:37:6B:60:07:42:48:90:45:34:B1:14:82:D5:BF:0E:69:8E:CC:49:8D:F5:25:77:EB:F2:E9:3B:9
Alias name: affirmtrustpremiumecc
  SHA1: B8:23:6B:00:2F:1D:16:86:53:01:55:6C:11:A4:37:CA:EB:FF:C3:BB
  SHA256:
 BD:71:FD:F6:DA:97:E4:CF:62:D1:64:7A:DD:25:81:B0:7D:79:AD:F8:39:7E:B4:EC:BA:9C:5E:84:88:82:14:2
Alias name: affirmtrustpremiumeccca
  SHA1: B8:23:6B:00:2F:1D:16:86:53:01:55:6C:11:A4:37:CA:EB:FF:C3:BB
  SHA256:
 BD:71:FD:F6:DA:97:E4:CF:62:D1:64:7A:DD:25:81:B0:7D:79:AD:F8:39:7E:B4:EC:BA:9C:5E:84:88:82:14:2
Alias name: amazon-ca-g4-acm1
  SHA1: F2:0D:28:B6:29:C2:2C:5E:84:05:E6:02:4D:97:FE:8F:A0:84:93:A0
  SHA256:
 B0:11:A4:F7:29:6C:74:D8:2B:F5:62:DF:87:D7:28:C7:1F:B5:8C:F4:E6:73:F2:78:FC:DA:F3:FF:83:A6:8C:8
Alias name: amazon-ca-g4-acm2
  SHA1: A7:E6:45:32:1F:7A:B7:AD:C0:70:EA:73:5F:AB:ED:C3:DA:B4:D0:C8
  SHA256:
 D7:A8:7C:69:95:D0:E2:04:2A:32:70:A7:E2:87:FE:A7:E8:F4:C1:70:62:F7:90:C3:EB:BB:53:F2:AC:39:26:B
Alias name: amazon-ca-g4-acm3
  SHA1: 7A:DB:56:57:5F:D6:EE:67:85:0A:64:BB:1C:E9:E4:B0:9A:DB:9D:07
  SHA256:
 6B:EB:9D:20:2E:C2:00:70:BD:D2:5E:D3:C0:C8:33:2C:B4:78:07:C5:82:94:4E:7E:23:28:22:71:A4:8E:0E:C
Alias name: amazon-ca-g4-legacy
  SHA1: EA:E7:DE:F9:0A:BE:9F:0B:68:CE:B7:24:0D:80:74:03:BF:6E:B1:6E
  SHA256:
 CD:72:C4:7F:B4:AD:28:A4:67:2B:E1:86:47:D4:40:E9:3B:16:2D:95:DB:3C:2F:94:BB:81:D9:09:F7:91:24:5
Alias name: amazon-root-ca-ecc-384-1
  SHA1: F9:5E:4A:AB:9C:2D:57:61:63:3D:B2:57:B4:0F:24:9E:7B:E2:23:7D
  SHA256:
 C6:BD:E5:66:C2:72:2A:0E:96:E9:C1:2C:BF:38:92:D9:55:4D:29:03:57:30:72:40:7F:4E:70:17:3B:3C:9B:6
Alias name: amazon-root-ca-rsa-2k-1
```

```
  SHA1: 8A:9A:AC:27:FC:86:D4:50:23:AD:D5:63:F9:1E:AE:2C:AF:63:08:6C
  SHA256:
 0F:8F:33:83:FB:70:02:89:49:24:E1:AA:B0:D7:FB:5A:BF:98:DF:75:8E:0F:FE:61:86:92:BC:F0:75:35:CC:8
Alias name: amazon-root-ca-rsa-4k-1
  SHA1: EC:BD:09:61:F5:7A:B6:A8:76:BB:20:8F:14:05:ED:7E:70:ED:39:45
  SHA256:
 36:AE:AD:C2:6A:60:07:90:6B:83:A3:73:2D:D1:2B:D4:00:5E:C7:F2:76:11:99:A9:D4:DA:63:2F:59:B2:8B:C
Alias name: amazon1
  SHA1: 8D:A7:F9:65:EC:5E:FC:37:91:0F:1C:6E:59:FD:C1:CC:6A:6E:DE:16
  SHA256:
 8E:CD:E6:88:4F:3D:87:B1:12:5B:A3:1A:C3:FC:B1:3D:70:16:DE:7F:57:CC:90:4F:E1:CB:97:C6:AE:98:19:6
Alias name: amazon2
  SHA1: 5A:8C:EF:45:D7:A6:98:59:76:7A:8C:8B:44:96:B5:78:CF:47:4B:1A
  SHA256:
 1B:A5:B2:AA:8C:65:40:1A:82:96:01:18:F8:0B:EC:4F:62:30:4D:83:CE:C4:71:3A:19:C3:9C:01:1E:A4:6D:B
Alias name: amazon3
  SHA1: 0D:44:DD:8C:3C:8C:1A:1A:58:75:64:81:E9:0F:2E:2A:FF:B3:D2:6E
  SHA256:
 18:CE:6C:FE:7B:F1:4E:60:B2:E3:47:B8:DF:E8:68:CB:31:D0:2E:BB:3A:DA:27:15:69:F5:03:43:B4:6D:B3:A
Alias name: amazon4
  SHA1: F6:10:84:07:D6:F8:BB:67:98:0C:C2:E2:44:C2:EB:AE:1C:EF:63:BE
  SHA256:
 E3:5D:28:41:9E:D0:20:25:CF:A6:90:38:CD:62:39:62:45:8D:A5:C6:95:FB:DE:A3:C2:2B:0B:FB:25:89:70:9
Alias name: amazonrootca1
  SHA1: 8D:A7:F9:65:EC:5E:FC:37:91:0F:1C:6E:59:FD:C1:CC:6A:6E:DE:16
  SHA256:
 8E:CD:E6:88:4F:3D:87:B1:12:5B:A3:1A:C3:FC:B1:3D:70:16:DE:7F:57:CC:90:4F:E1:CB:97:C6:AE:98:19:6
Alias name: amazonrootca2
  SHA1: 5A:8C:EF:45:D7:A6:98:59:76:7A:8C:8B:44:96:B5:78:CF:47:4B:1A
  SHA256:
 1B:A5:B2:AA:8C:65:40:1A:82:96:01:18:F8:0B:EC:4F:62:30:4D:83:CE:C4:71:3A:19:C3:9C:01:1E:A4:6D:B
Alias name: amazonrootca3
  SHA1: 0D:44:DD:8C:3C:8C:1A:1A:58:75:64:81:E9:0F:2E:2A:FF:B3:D2:6E
  SHA256:
 18:CE:6C:FE:7B:F1:4E:60:B2:E3:47:B8:DF:E8:68:CB:31:D0:2E:BB:3A:DA:27:15:69:F5:03:43:B4:6D:B3:A
Alias name: amazonrootca4
  SHA1: F6:10:84:07:D6:F8:BB:67:98:0C:C2:E2:44:C2:EB:AE:1C:EF:63:BE
  SHA256:
 E3:5D:28:41:9E:D0:20:25:CF:A6:90:38:CD:62:39:62:45:8D:A5:C6:95:FB:DE:A3:C2:2B:0B:FB:25:89:70:9
Alias name: amzninternalinfoseccag3
  SHA1: B9:B1:CA:38:F7:BF:9C:D2:D4:95:E7:B6:5E:75:32:9B:A8:78:2E:F6
  SHA256:
 81:03:0B:C7:E2:54:DA:7B:F8:B7:45:DB:DD:41:15:89:B5:A3:81:86:FB:4B:29:77:1F:84:0A:18:D9:67:6D:6
Alias name: amzninternalrootca
```

```
  SHA1: A7:B7:F6:15:8A:FF:1E:C8:85:13:38:BC:93:EB:A2:AB:A4:09:EF:06
  SHA256:
 0E:DE:63:C1:DC:7A:8E:11:F1:AB:BC:05:4F:59:EE:49:9D:62:9A:2F:DE:9C:A7:16:32:A2:64:29:3E:8B:66:A
Alias name: aolrootca1
  SHA1: 39:21:C1:15:C1:5D:0E:CA:5C:CB:5B:C4:F0:7D:21:D8:05:0B:56:6A
  SHA256:
 77:40:73:12:C6:3A:15:3D:5B:C0:0B:4E:51:75:9C:DF:DA:C2:37:DC:2A:33:B6:79:46:E9:8E:9B:FA:68:0A:E
Alias name: aolrootca2
  SHA1: 85:B5:FF:67:9B:0C:79:96:1F:C8:6E:44:22:00:46:13:DB:17:92:84
  SHA256:
 7D:3B:46:5A:60:14:E5:26:C0:AF:FC:EE:21:27:D2:31:17:27:AD:81:1C:26:84:2D:00:6A:F3:73:06:CC:80:B
Alias name: atostrustedroot2011
  SHA1: 2B:B1:F5:3E:55:0C:1D:C5:F1:D4:E6:B7:6A:46:4B:55:06:02:AC:21
  SHA256:
 F3:56:BE:A2:44:B7:A9:1E:B3:5D:53:CA:9A:D7:86:4A:CE:01:8E:2D:35:D5:F8:F9:6D:DF:68:A6:F4:1A:A4:7
Alias name: autoridaddecertificacionfirmaprofesionalcifa62634068
  SHA1: AE:C5:FB:3F:C8:E1:BF:C4:E5:4F:03:07:5A:9A:E8:00:B7:F7:B6:FA
  SHA256:
 04:04:80:28:BF:1F:28:64:D4:8F:9A:D4:D8:32:94:36:6A:82:88:56:55:3F:3B:14:30:3F:90:14:7F:5D:40:E
Alias name: baltimorecodesigningca
  SHA1: 30:46:D8:C8:88:FF:69:30:C3:4A:FC:CD:49:27:08:7C:60:56:7B:0D
  SHA256:
 A9:15:45:DB:D2:E1:9C:4C:CD:F9:09:AA:71:90:0D:18:C7:35:1C:89:B3:15:F0:F1:3D:05:C1:3A:8F:FB:46:8
Alias name: baltimorecybertrustca
  SHA1: D4:DE:20:D0:5E:66:FC:53:FE:1A:50:88:2C:78:DB:28:52:CA:E4:74
  SHA256:
 16:AF:57:A9:F6:76:B0:AB:12:60:95:AA:5E:BA:DE:F2:2A:B3:11:19:D6:44:AC:95:CD:4B:93:DB:F3:F2:6A:E
Alias name: baltimorecybertrustroot
  SHA1: D4:DE:20:D0:5E:66:FC:53:FE:1A:50:88:2C:78:DB:28:52:CA:E4:74
  SHA256:
 16:AF:57:A9:F6:76:B0:AB:12:60:95:AA:5E:BA:DE:F2:2A:B3:11:19:D6:44:AC:95:CD:4B:93:DB:F3:F2:6A:E
Alias name: buypassclass2ca
  SHA1: 49:0A:75:74:DE:87:0A:47:FE:58:EE:F6:C7:6B:EB:C6:0B:12:40:99
  SHA256:
 9A:11:40:25:19:7C:5B:B9:5D:94:E6:3D:55:CD:43:79:08:47:B6:46:B2:3C:DF:11:AD:A4:A0:0E:FF:15:FB:4
Alias name: buypassclass2rootca
  SHA1: 49:0A:75:74:DE:87:0A:47:FE:58:EE:F6:C7:6B:EB:C6:0B:12:40:99
  SHA256:
 9A:11:40:25:19:7C:5B:B9:5D:94:E6:3D:55:CD:43:79:08:47:B6:46:B2:3C:DF:11:AD:A4:A0:0E:FF:15:FB:4
Alias name: buypassclass3ca
  SHA1: DA:FA:F7:FA:66:84:EC:06:8F:14:50:BD:C7:C2:81:A5:BC:A9:64:57
  SHA256:
 ED:F7:EB:BC:A2:7A:2A:38:4D:38:7B:7D:40:10:C6:66:E2:ED:B4:84:3E:4C:29:B4:AE:1D:5B:93:32:E6:B2:4
Alias name: buypassclass3rootca
```

```
  SHA1: DA:FA:F7:FA:66:84:EC:06:8F:14:50:BD:C7:C2:81:A5:BC:A9:64:57
  SHA256:
 ED:F7:EB:BC:A2:7A:2A:38:4D:38:7B:7D:40:10:C6:66:E2:ED:B4:84:3E:4C:29:B4:AE:1D:5B:93:32:E6:B2:4
Alias name: cadisigrootr2
  SHA1: B5:61:EB:EA:A4:DE:E4:25:4B:69:1A:98:A5:57:47:C2:34:C7:D9:71
  SHA256:
 E2:3D:4A:03:6D:7B:70:E9:F5:95:B1:42:20:79:D2:B9:1E:DF:BB:1F:B6:51:A0:63:3E:AA:8A:9D:C5:F8:07:0
Alias name: camerfirmachambersca
  SHA1: 78:6A:74:AC:76:AB:14:7F:9C:6A:30:50:BA:9E:A8:7E:FE:9A:CE:3C
  SHA256:
 06:3E:4A:FA:C4:91:DF:D3:32:F3:08:9B:85:42:E9:46:17:D8:93:D7:FE:94:4E:10:A7:93:7E:E2:9D:96:93:C
Alias name: camerfirmachamberscommerceca
  SHA1: 6E:3A:55:A4:19:0C:19:5C:93:84:3C:C0:DB:72:2E:31:30:61:F0:B1
  SHA256:
 0C:25:8A:12:A5:67:4A:EF:25:F2:8B:A7:DC:FA:EC:EE:A3:48:E5:41:E6:F5:CC:4E:E6:3B:71:B3:61:60:6A:C
Alias name: camerfirmachambersignca
  SHA1: 4A:BD:EE:EC:95:0D:35:9C:89:AE:C7:52:A1:2C:5B:29:F6:D6:AA:0C
  SHA256:
 13:63:35:43:93:34:A7:69:80:16:A0:D3:24:DE:72:28:4E:07:9D:7B:52:20:BB:8F:BD:74:78:16:EE:BE:BA:C
Alias name: certigna
  SHA1: B1:2E:13:63:45:86:A4:6F:1A:B2:60:68:37:58:2D:C4:AC:FD:94:97
  SHA256:
 E3:B6:A2:DB:2E:D7:CE:48:84:2F:7A:C5:32:41:C7:B7:1D:54:14:4B:FB:40:C1:1F:3F:1D:0B:42:F5:EE:A1:2
Alias name: certignarootca
  SHA1: 2D:0D:52:14:FF:9E:AD:99:24:01:74:20:47:6E:6C:85:27:27:F5:43
  SHA256:
 D4:8D:3D:23:EE:DB:50:A4:59:E5:51:97:60:1C:27:77:4B:9D:7B:18:C9:4D:5A:05:95:11:A1:02:50:B9:31:6
Alias name: certplusclass2primaryca
  SHA1: 74:20:74:41:72:9C:DD:92:EC:79:31:D8:23:10:8D:C2:81:92:E2:BB
  SHA256:
 0F:99:3C:8A:EF:97:BA:AF:56:87:14:0E:D5:9A:D1:82:1B:B4:AF:AC:F0:AA:9A:58:B5:D5:7A:33:8A:3A:FB:C
Alias name: certplusclass3pprimaryca
  SHA1: 21:6B:2A:29:E6:2A:00:CE:82:01:46:D8:24:41:41:B9:25:11:B2:79
  SHA256:
 CC:C8:94:89:37:1B:AD:11:1C:90:61:9B:EA:24:0A:2E:6D:AD:D9:9F:9F:6E:1D:4D:41:E5:8E:D6:DE:3D:02:8
Alias name: certsignrootca
  SHA1: FA:B7:EE:36:97:26:62:FB:2D:B0:2A:F6:BF:03:FD:E8:7C:4B:2F:9B
  SHA256:
 EA:A9:62:C4:FA:4A:6B:AF:EB:E4:15:19:6D:35:1C:CD:88:8D:4F:53:F3:FA:8A:E6:D7:C4:66:A9:4E:60:42:B
Alias name: certsignrootcag2
  SHA1: 26:F9:93:B4:ED:3D:28:27:B0:B9:4B:A7:E9:15:1D:A3:8D:92:E5:32
  SHA256:
 65:7C:FE:2F:A7:3F:AA:38:46:25:71:F3:32:A2:36:3A:46:FC:E7:02:09:51:71:07:02:CD:FB:B6:EE:DA:33:0
Alias name: certum2
```

```
  SHA1: D3:DD:48:3E:2B:BF:4C:05:E8:AF:10:F5:FA:76:26:CF:D3:DC:30:92
  SHA256:
 B6:76:F2:ED:DA:E8:77:5C:D3:6C:B0:F6:3C:D1:D4:60:39:61:F4:9E:62:65:BA:01:3A:2F:03:07:B6:D0:B8:0
Alias name: certumca
  SHA1: 62:52:DC:40:F7:11:43:A2:2F:DE:9E:F7:34:8E:06:42:51:B1:81:18
  SHA256:
 D8:E0:FE:BC:1D:B2:E3:8D:00:94:0F:37:D2:7D:41:34:4D:99:3E:73:4B:99:D5:65:6D:97:78:D4:D8:14:36:2
Alias name: certumtrustednetworkca
  SHA1: 07:E0:32:E0:20:B7:2C:3F:19:2F:06:28:A2:59:3A:19:A7:0F:06:9E
  SHA256:
 5C:58:46:8D:55:F5:8E:49:7E:74:39:82:D2:B5:00:10:B6:D1:65:37:4A:CF:83:A7:D4:A3:2D:B7:68:C4:40:8
Alias name: certumtrustednetworkca2
  SHA1: D3:DD:48:3E:2B:BF:4C:05:E8:AF:10:F5:FA:76:26:CF:D3:DC:30:92
  SHA256:
 B6:76:F2:ED:DA:E8:77:5C:D3:6C:B0:F6:3C:D1:D4:60:39:61:F4:9E:62:65:BA:01:3A:2F:03:07:B6:D0:B8:0
Alias name: cfcaevroot
  SHA1: E2:B8:29:4B:55:84:AB:6B:58:C2:90:46:6C:AC:3F:B8:39:8F:84:83
  SHA256:
 5C:C3:D7:8E:4E:1D:5E:45:54:7A:04:E6:87:3E:64:F9:0C:F9:53:6D:1C:CC:2E:F8:00:F3:55:C4:C5:FD:70:F
Alias name: chambersofcommerceroot2008
  SHA1: 78:6A:74:AC:76:AB:14:7F:9C:6A:30:50:BA:9E:A8:7E:FE:9A:CE:3C
  SHA256:
 06:3E:4A:FA:C4:91:DF:D3:32:F3:08:9B:85:42:E9:46:17:D8:93:D7:FE:94:4E:10:A7:93:7E:E2:9D:96:93:C
Alias name: chunghwaepkirootca
  SHA1: 67:65:0D:F1:7E:8E:7E:5B:82:40:A4:F4:56:4B:CF:E2:3D:69:C6:F0
  SHA256:
 C0:A6:F4:DC:63:A2:4B:FD:CF:54:EF:2A:6A:08:2A:0A:72:DE:35:80:3E:2F:F5:FF:52:7A:E5:D8:72:06:DF:D
Alias name: cia-crt-g3-01-ca
  SHA1: 2B:EE:2C:BA:A3:1D:B5:FE:60:40:41:95:08:ED:46:82:39:4D:ED:E2
  SHA256:
 20:48:AD:4C:EC:90:7F:FA:4A:15:D4:CE:45:E3:C8:E4:2C:EA:78:33:DC:C7:D3:40:48:FC:60:47:27:42:99:E
Alias name: cia-crt-g3-02-ca
  SHA1: 96:4A:BB:A7:BD:DA:FC:97:34:C0:0A:2D:F0:05:98:F7:E6:C6:6F:09
  SHA256:
 93:F1:72:FB:BA:43:31:5C:06:EE:0F:9F:04:89:B8:F6:88:BC:75:15:3C:BE:B4:80:AC:A7:14:3A:F6:FC:4A:C
Alias name: comodo-ca
  SHA1: AF:E5:D2:44:A8:D1:19:42:30:FF:47:9F:E2:F8:97:BB:CD:7A:8C:B4
  SHA256:
 52:F0:E1:C4:E5:8E:C6:29:29:1B:60:31:7F:07:46:71:B8:5D:7E:A8:0D:5B:07:27:34:63:53:4B:32:B4:02:3
Alias name: comodoaaaca
  SHA1: D1:EB:23:A4:6D:17:D6:8F:D9:25:64:C2:F1:F1:60:17:64:D8:E3:49
  SHA256:
 D7:A7:A0:FB:5D:7E:27:31:D7:71:E9:48:4E:BC:DE:F7:1D:5F:0C:3E:0A:29:48:78:2B:C8:3E:E0:EA:69:9E:F
Alias name: comodoaaaservicesroot
```

```
  SHA1: D1:EB:23:A4:6D:17:D6:8F:D9:25:64:C2:F1:F1:60:17:64:D8:E3:49
  SHA256:
 D7:A7:A0:FB:5D:7E:27:31:D7:71:E9:48:4E:BC:DE:F7:1D:5F:0C:3E:0A:29:48:78:2B:C8:3E:E0:EA:69:9E:F
Alias name: comodocertificationauthority
  SHA1: 66:31:BF:9E:F7:4F:9E:B6:C9:D5:A6:0C:BA:6A:BE:D1:F7:BD:EF:7B
  SHA256:
 0C:2C:D6:3D:F7:80:6F:A3:99:ED:E8:09:11:6B:57:5B:F8:79:89:F0:65:18:F9:80:8C:86:05:03:17:8B:AF:6
Alias name: comodoecccertificationauthority
  SHA1: 9F:74:4E:9F:2B:4D:BA:EC:0F:31:2C:50:B6:56:3B:8E:2D:93:C3:11
  SHA256:
 17:93:92:7A:06:14:54:97:89:AD:CE:2F:8F:34:F7:F0:B6:6D:0F:3A:E3:A3:B8:4D:21:EC:15:DB:BA:4F:AD:C
Alias name: comodorsacertificationauthority
  SHA1: AF:E5:D2:44:A8:D1:19:42:30:FF:47:9F:E2:F8:97:BB:CD:7A:8C:B4
  SHA256:
 52:F0:E1:C4:E5:8E:C6:29:29:1B:60:31:7F:07:46:71:B8:5D:7E:A8:0D:5B:07:27:34:63:53:4B:32:B4:02:3
Alias name: cybertrustglobalroot
  SHA1: 5F:43:E5:B1:BF:F8:78:8C:AC:1C:C7:CA:4A:9A:C6:22:2B:CC:34:C6
  SHA256:
 96:0A:DF:00:63:E9:63:56:75:0C:29:65:DD:0A:08:67:DA:0B:9C:BD:6E:77:71:4A:EA:FB:23:49:AB:39:3D:A
Alias name: deprecateditsecca
  SHA1: 12:12:0B:03:0E:15:14:54:F4:DD:B3:F5:DE:13:6E:83:5A:29:72:9D
  SHA256:
 9A:59:DA:86:24:1A:FD:BA:A3:39:FA:9C:FD:21:6A:0B:06:69:4D:E3:7E:37:52:6B:BE:63:C8:BC:83:74:2E:C
Alias name: deutschetelekomrootca2
  SHA1: 85:A4:08:C0:9C:19:3E:5D:51:58:7D:CD:D6:13:30:FD:8C:DE:37:BF
  SHA256:
 B6:19:1A:50:D0:C3:97:7F:7D:A9:9B:CD:AA:C8:6A:22:7D:AE:B9:67:9E:C7:0B:A3:B0:C9:D9:22:71:C1:70:D
Alias name: digicertassuredidrootca
  SHA1: 05:63:B8:63:0D:62:D7:5A:BB:C8:AB:1E:4B:DF:B5:A8:99:B2:4D:43
  SHA256:
 3E:90:99:B5:01:5E:8F:48:6C:00:BC:EA:9D:11:1E:E7:21:FA:BA:35:5A:89:BC:F1:DF:69:56:1E:3D:C6:32:5
Alias name: digicertassuredidrootg2
  SHA1: A1:4B:48:D9:43:EE:0A:0E:40:90:4F:3C:E0:A4:C0:91:93:51:5D:3F
  SHA256:
 7D:05:EB:B6:82:33:9F:8C:94:51:EE:09:4E:EB:FE:FA:79:53:A1:14:ED:B2:F4:49:49:45:2F:AB:7D:2F:C1:8
Alias name: digicertassuredidrootg3
  SHA1: F5:17:A2:4F:9A:48:C6:C9:F8:A2:00:26:9F:DC:0F:48:2C:AB:30:89
  SHA256:
 7E:37:CB:8B:4C:47:09:0C:AB:36:55:1B:A6:F4:5D:B8:40:68:0F:BA:16:6A:95:2D:B1:00:71:7F:43:05:3F:C
Alias name: digicertglobalrootca
  SHA1: A8:98:5D:3A:65:E5:E5:C4:B2:D7:D6:6D:40:C6:DD:2F:B1:9C:54:36
  SHA256:
 43:48:A0:E9:44:4C:78:CB:26:5E:05:8D:5E:89:44:B4:D8:4F:96:62:BD:26:DB:25:7F:89:34:A4:43:C7:01:6
Alias name: digicertglobalrootg2
```

```
  SHA1: DF:3C:24:F9:BF:D6:66:76:1B:26:80:73:FE:06:D1:CC:8D:4F:82:A4
  SHA256:
 CB:3C:CB:B7:60:31:E5:E0:13:8F:8D:D3:9A:23:F9:DE:47:FF:C3:5E:43:C1:14:4C:EA:27:D4:6A:5A:B1:CB:5
Alias name: digicertglobalrootg3
  SHA1: 7E:04:DE:89:6A:3E:66:6D:00:E6:87:D3:3F:FA:D9:3B:E8:3D:34:9E
  SHA256:
 31:AD:66:48:F8:10:41:38:C7:38:F3:9E:A4:32:01:33:39:3E:3A:18:CC:02:29:6E:F9:7C:2A:C9:EF:67:31:D
Alias name: digicerthighassuranceevrootca
  SHA1: 5F:B7:EE:06:33:E2:59:DB:AD:0C:4C:9A:E6:D3:8F:1A:61:C7:DC:25
  SHA256:
 74:31:E5:F4:C3:C1:CE:46:90:77:4F:0B:61:E0:54:40:88:3B:A9:A0:1E:D0:0B:A6:AB:D7:80:6E:D3:B1:18:C
Alias name: digicerttrustedrootg4
  SHA1: DD:FB:16:CD:49:31:C9:73:A2:03:7D:3F:C8:3A:4D:7D:77:5D:05:E4
  SHA256:
 55:2F:7B:DC:F1:A7:AF:9E:6C:E6:72:01:7F:4F:12:AB:F7:72:40:C7:8E:76:1A:C2:03:D1:D9:D2:0A:C8:99:8
Alias name: dstrootcax3
  SHA1: DA:C9:02:4F:54:D8:F6:DF:94:93:5F:B1:73:26:38:CA:6A:D7:7C:13
  SHA256:
 06:87:26:03:31:A7:24:03:D9:09:F1:05:E6:9B:CF:0D:32:E1:BD:24:93:FF:C6:D9:20:6D:11:BC:D6:77:07:3
Alias name: dtrustrootclass3ca22009
  SHA1: 58:E8:AB:B0:36:15:33:FB:80:F7:9B:1B:6D:29:D3:FF:8D:5F:00:F0
  SHA256:
 49:E7:A4:42:AC:F0:EA:62:87:05:00:54:B5:25:64:B6:50:E4:F4:9E:42:E3:48:D6:AA:38:E0:39:E9:57:B1:C
Alias name: dtrustrootclass3ca2ev2009
  SHA1: 96:C9:1B:0B:95:B4:10:98:42:FA:D0:D8:22:79:FE:60:FA:B9:16:83
  SHA256:
 EE:C5:49:6B:98:8C:E9:86:25:B9:34:09:2E:EC:29:08:BE:D0:B0:F3:16:C2:D4:73:0C:84:EA:F1:F3:D3:48:8
Alias name: ecacc
  SHA1: 28:90:3A:63:5B:52:80:FA:E6:77:4C:0B:6D:A7:D6:BA:A6:4A:F2:E8
  SHA256:
 88:49:7F:01:60:2F:31:54:24:6A:E2:8C:4D:5A:EF:10:F1:D8:7E:BB:76:62:6F:4A:E0:B7:F9:5B:A7:96:87:9
Alias name: emsigneccrootcac3
  SHA1: B6:AF:43:C2:9B:81:53:7D:F6:EF:6B:C3:1F:1F:60:15:0C:EE:48:66
  SHA256:
 BC:4D:80:9B:15:18:9D:78:DB:3E:1D:8C:F4:F9:72:6A:79:5D:A1:64:3C:A5:F1:35:8E:1D:DB:0E:DC:0D:7E:B
Alias name: emsigneccrootcag3
  SHA1: 30:43:FA:4F:F2:57:DC:A0:C3:80:EE:2E:58:EA:78:B2:3F:E6:BB:C1
  SHA256:
 86:A1:EC:BA:08:9C:4A:8D:3B:BE:27:34:C6:12:BA:34:1D:81:3E:04:3C:F9:E8:A8:62:CD:5C:57:A3:6B:BE:6
Alias name: emsignrootcac1
  SHA1: E7:2E:F1:DF:FC:B2:09:28:CF:5D:D4:D5:67:37:B1:51:CB:86:4F:01
  SHA256:
 12:56:09:AA:30:1D:A0:A2:49:B9:7A:82:39:CB:6A:34:21:6F:44:DC:AC:9F:39:54:B1:42:92:F2:E8:C8:60:8
Alias name: emsignrootcag1
```

```
  SHA1: 8A:C7:AD:8F:73:AC:4E:C1:B5:75:4D:A5:40:F4:FC:CF:7C:B5:8E:8C
  SHA256:
 40:F6:AF:03:46:A9:9A:A1:CD:1D:55:5A:4E:9C:CE:62:C7:F9:63:46:03:EE:40:66:15:83:3D:C8:C8:D0:03:6
Alias name: entrust2048ca
  SHA1: 50:30:06:09:1D:97:D4:F5:AE:39:F7:CB:E7:92:7D:7D:65:2D:34:31
  SHA256:
 6D:C4:71:72:E0:1C:BC:B0:BF:62:58:0D:89:5F:E2:B8:AC:9A:D4:F8:73:80:1E:0C:10:B9:C8:37:D2:1E:B1:7
Alias name: entrustevca
  SHA1: B3:1E:B1:B7:40:E3:6C:84:02:DA:DC:37:D4:4D:F5:D4:67:49:52:F9
  SHA256:
 73:C1:76:43:4F:1B:C6:D5:AD:F4:5B:0E:76:E7:27:28:7C:8D:E5:76:16:C1:E6:E6:14:1A:2B:2C:BC:7D:8E:4
Alias name: entrustnetpremium2048secureserverca
  SHA1: 50:30:06:09:1D:97:D4:F5:AE:39:F7:CB:E7:92:7D:7D:65:2D:34:31
  SHA256:
 6D:C4:71:72:E0:1C:BC:B0:BF:62:58:0D:89:5F:E2:B8:AC:9A:D4:F8:73:80:1E:0C:10:B9:C8:37:D2:1E:B1:7
Alias name: entrustrootcag2
  SHA1: 8C:F4:27:FD:79:0C:3A:D1:66:06:8D:E8:1E:57:EF:BB:93:22:72:D4
  SHA256:
 43:DF:57:74:B0:3E:7F:EF:5F:E4:0D:93:1A:7B:ED:F1:BB:2E:6B:42:73:8C:4E:6D:38:41:10:3D:3A:A7:F3:3
Alias name: entrustrootcertificationauthority
  SHA1: B3:1E:B1:B7:40:E3:6C:84:02:DA:DC:37:D4:4D:F5:D4:67:49:52:F9
  SHA256:
 73:C1:76:43:4F:1B:C6:D5:AD:F4:5B:0E:76:E7:27:28:7C:8D:E5:76:16:C1:E6:E6:14:1A:2B:2C:BC:7D:8E:4
Alias name: entrustrootcertificationauthorityec1
  SHA1: 20:D8:06:40:DF:9B:25:F5:12:25:3A:11:EA:F7:59:8A:EB:14:B5:47
  SHA256:
 02:ED:0E:B2:8C:14:DA:45:16:5C:56:67:91:70:0D:64:51:D7:FB:56:F0:B2:AB:1D:3B:8E:B0:70:E5:6E:DF:F
Alias name: entrustrootcertificationauthorityg2
  SHA1: 8C:F4:27:FD:79:0C:3A:D1:66:06:8D:E8:1E:57:EF:BB:93:22:72:D4
  SHA256:
 43:DF:57:74:B0:3E:7F:EF:5F:E4:0D:93:1A:7B:ED:F1:BB:2E:6B:42:73:8C:4E:6D:38:41:10:3D:3A:A7:F3:3
Alias name: entrustrootcertificationauthorityg4
  SHA1: 14:88:4E:86:26:37:B0:26:AF:59:62:5C:40:77:EC:35:29:BA:96:01
  SHA256:
 DB:35:17:D1:F6:73:2A:2D:5A:B9:7C:53:3E:C7:07:79:EE:32:70:A6:2F:B4:AC:42:38:37:24:60:E6:F0:1E:8
Alias name: epkirootcertificationauthority
  SHA1: 67:65:0D:F1:7E:8E:7E:5B:82:40:A4:F4:56:4B:CF:E2:3D:69:C6:F0
  SHA256:
 C0:A6:F4:DC:63:A2:4B:FD:CF:54:EF:2A:6A:08:2A:0A:72:DE:35:80:3E:2F:F5:FF:52:7A:E5:D8:72:06:DF:D
Alias name: equifaxsecureebusinessca1
  SHA1: AE:E6:3D:70:E3:76:FB:C7:3A:EB:B0:A1:C1:D4:C4:7A:A7:40:B3:F4
  SHA256:
 2E:3A:2B:B5:11:25:05:83:6C:A8:96:8B:E2:CB:37:27:CE:9B:56:84:5C:6E:E9:8E:91:85:10:4A:FB:9A:F5:9
Alias name: equifaxsecureglobalebusinessca1
```

```
  SHA1: 3A:74:CB:7A:47:DB:70:DE:89:1F:24:35:98:64:B8:2D:82:BD:1A:36
  SHA256:
 86:AB:5A:65:71:D3:32:9A:BC:D2:E4:E6:37:66:8B:A8:9C:73:1E:C2:93:B6:CB:A6:0F:71:63:40:A0:91:CE:A
Alias name: eszignorootca2017
  SHA1: 89:D4:83:03:4F:9E:9A:48:80:5F:72:37:D4:A9:A6:EF:CB:7C:1F:D1
  SHA256:
 BE:B0:0B:30:83:9B:9B:C3:2C:32:E4:44:79:05:95:06:41:F2:64:21:B1:5E:D0:89:19:8B:51:8A:E2:EA:1B:9
Alias name: etugracertificationauthority
  SHA1: 51:C6:E7:08:49:06:6E:F3:92:D4:5C:A0:0D:6D:A3:62:8F:C3:52:39
  SHA256:
 B0:BF:D5:2B:B0:D7:D9:BD:92:BF:5D:4D:C1:3D:A2:55:C0:2C:54:2F:37:83:65:EA:89:39:11:F5:5E:55:F2:3
Alias name: gd-class2-root.pem
  SHA1: 27:96:BA:E6:3F:18:01:E2:77:26:1B:A0:D7:77:70:02:8F:20:EE:E4
  SHA256:
 C3:84:6B:F2:4B:9E:93:CA:64:27:4C:0E:C6:7C:1E:CC:5E:02:4F:FC:AC:D2:D7:40:19:35:0E:81:FE:54:6A:E
Alias name: gd_bundle-g2.pem
  SHA1: 27:AC:93:69:FA:F2:52:07:BB:26:27:CE:FA:CC:BE:4E:F9:C3:19:B8
  SHA256:
 97:3A:41:27:6F:FD:01:E0:27:A2:AA:D4:9E:34:C3:78:46:D3:E9:76:FF:6A:62:0B:67:12:E3:38:32:04:1A:A
Alias name: gdcatrustauthr5root
  SHA1: 0F:36:38:5B:81:1A:25:C3:9B:31:4E:83:CA:E9:34:66:70:CC:74:B4
  SHA256:
 BF:FF:8F:D0:44:33:48:7D:6A:8A:A6:0C:1A:29:76:7A:9F:C2:BB:B0:5E:42:0F:71:3A:13:B9:92:89:1D:38:9
Alias name: gdroot-g2.pem
  SHA1: 47:BE:AB:C9:22:EA:E8:0E:78:78:34:62:A7:9F:45:C2:54:FD:E6:8B
  SHA256:
 45:14:0B:32:47:EB:9C:C8:C5:B4:F0:D7:B5:30:91:F7:32:92:08:9E:6E:5A:63:E2:74:9D:D3:AC:A9:19:8E:D
Alias name: geotrustglobalca
  SHA1: DE:28:F4:A4:FF:E5:B9:2F:A3:C5:03:D1:A3:49:A7:F9:96:2A:82:12
  SHA256:
 FF:85:6A:2D:25:1D:CD:88:D3:66:56:F4:50:12:67:98:CF:AB:AA:DE:40:79:9C:72:2D:E4:D2:B5:DB:36:A7:3
Alias name: geotrustprimaryca
  SHA1: 32:3C:11:8E:1B:F7:B8:B6:52:54:E2:E2:10:0D:D6:02:90:37:F0:96
  SHA256:
 37:D5:10:06:C5:12:EA:AB:62:64:21:F1:EC:8C:92:01:3F:C5:F8:2A:E9:8E:E5:33:EB:46:19:B8:DE:B4:D0:6
Alias name: geotrustprimarycag2
  SHA1: 8D:17:84:D5:37:F3:03:7D:EC:70:FE:57:8B:51:9A:99:E6:10:D7:B0
  SHA256:
 5E:DB:7A:C4:3B:82:A0:6A:87:61:E8:D7:BE:49:79:EB:F2:61:1F:7D:D7:9B:F9:1C:1C:6B:56:6A:21:9E:D7:6
Alias name: geotrustprimarycag3
  SHA1: 03:9E:ED:B8:0B:E7:A0:3C:69:53:89:3B:20:D2:D9:32:3A:4C:2A:FD
  SHA256:
 B4:78:B8:12:25:0D:F8:78:63:5C:2A:A7:EC:7D:15:5E:AA:62:5E:E8:29:16:E2:CD:29:43:61:88:6C:D1:FB:D
Alias name: geotrustprimarycertificationauthority
```

```
  SHA1: 32:3C:11:8E:1B:F7:B8:B6:52:54:E2:E2:10:0D:D6:02:90:37:F0:96
  SHA256:
 37:D5:10:06:C5:12:EA:AB:62:64:21:F1:EC:8C:92:01:3F:C5:F8:2A:E9:8E:E5:33:EB:46:19:B8:DE:B4:D0:6
Alias name: geotrustprimarycertificationauthorityg2
  SHA1: 8D:17:84:D5:37:F3:03:7D:EC:70:FE:57:8B:51:9A:99:E6:10:D7:B0
  SHA256:
 5E:DB:7A:C4:3B:82:A0:6A:87:61:E8:D7:BE:49:79:EB:F2:61:1F:7D:D7:9B:F9:1C:1C:6B:56:6A:21:9E:D7:6
Alias name: geotrustprimarycertificationauthorityg3
  SHA1: 03:9E:ED:B8:0B:E7:A0:3C:69:53:89:3B:20:D2:D9:32:3A:4C:2A:FD
  SHA256:
 B4:78:B8:12:25:0D:F8:78:63:5C:2A:A7:EC:7D:15:5E:AA:62:5E:E8:29:16:E2:CD:29:43:61:88:6C:D1:FB:D
Alias name: geotrustuniversalca
  SHA1: E6:21:F3:35:43:79:05:9A:4B:68:30:9D:8A:2F:74:22:15:87:EC:79
  SHA256:
 A0:45:9B:9F:63:B2:25:59:F5:FA:5D:4C:6D:B3:F9:F7:2F:F1:93:42:03:35:78:F0:73:BF:1D:1B:46:CB:B9:1
Alias name: geotrustuniversalca2
  SHA1: 37:9A:19:7B:41:85:45:35:0C:A6:03:69:F3:3C:2E:AF:47:4F:20:79
  SHA256:
 A0:23:4F:3B:C8:52:7C:A5:62:8E:EC:81:AD:5D:69:89:5D:A5:68:0D:C9:1D:1C:B8:47:7F:33:F8:78:B9:5B:0
Alias name: globalchambersignroot2008
  SHA1: 4A:BD:EE:EC:95:0D:35:9C:89:AE:C7:52:A1:2C:5B:29:F6:D6:AA:0C
  SHA256:
 13:63:35:43:93:34:A7:69:80:16:A0:D3:24:DE:72:28:4E:07:9D:7B:52:20:BB:8F:BD:74:78:16:EE:BE:BA:C
Alias name: globalsignca
  SHA1: B1:BC:96:8B:D4:F4:9D:62:2A:A8:9A:81:F2:15:01:52:A4:1D:82:9C
  SHA256:
 EB:D4:10:40:E4:BB:3E:C7:42:C9:E3:81:D3:1E:F2:A4:1A:48:B6:68:5C:96:E7:CE:F3:C1:DF:6C:D4:33:1C:9
Alias name: globalsigneccrootcar4
  SHA1: 69:69:56:2E:40:80:F4:24:A1:E7:19:9F:14:BA:F3:EE:58:AB:6A:BB
  SHA256:
 BE:C9:49:11:C2:95:56:76:DB:6C:0A:55:09:86:D7:6E:3B:A0:05:66:7C:44:2C:97:62:B4:FB:B7:73:DE:22:8
Alias name: globalsigneccrootcar5
  SHA1: 1F:24:C6:30:CD:A4:18:EF:20:69:FF:AD:4F:DD:5F:46:3A:1B:69:AA
  SHA256:
 17:9F:BC:14:8A:3D:D0:0F:D2:4E:A1:34:58:CC:43:BF:A7:F5:9C:81:82:D7:83:A5:13:F6:EB:EC:10:0C:89:2
Alias name: globalsignr2ca
  SHA1: 75:E0:AB:B6:13:85:12:27:1C:04:F8:5F:DD:DE:38:E4:B7:24:2E:FE
  SHA256:
 CA:42:DD:41:74:5F:D0:B8:1E:B9:02:36:2C:F9:D8:BF:71:9D:A1:BD:1B:1E:FC:94:6F:5B:4C:99:F4:2C:1B:9
Alias name: globalsignr3ca
  SHA1: D6:9B:56:11:48:F0:1C:77:C5:45:78:C1:09:26:DF:5B:85:69:76:AD
  SHA256:
 CB:B5:22:D7:B7:F1:27:AD:6A:01:13:86:5B:DF:1C:D4:10:2E:7D:07:59:AF:63:5A:7C:F4:72:0D:C9:63:C5:3
Alias name: globalsignrootca
```

```
  SHA1: B1:BC:96:8B:D4:F4:9D:62:2A:A8:9A:81:F2:15:01:52:A4:1D:82:9C
  SHA256:
 EB:D4:10:40:E4:BB:3E:C7:42:C9:E3:81:D3:1E:F2:A4:1A:48:B6:68:5C:96:E7:CE:F3:C1:DF:6C:D4:33:1C:9
Alias name: globalsignrootcar2
  SHA1: 75:E0:AB:B6:13:85:12:27:1C:04:F8:5F:DD:DE:38:E4:B7:24:2E:FE
  SHA256:
 CA:42:DD:41:74:5F:D0:B8:1E:B9:02:36:2C:F9:D8:BF:71:9D:A1:BD:1B:1E:FC:94:6F:5B:4C:99:F4:2C:1B:9
Alias name: globalsignrootcar3
  SHA1: D6:9B:56:11:48:F0:1C:77:C5:45:78:C1:09:26:DF:5B:85:69:76:AD
  SHA256:
 CB:B5:22:D7:B7:F1:27:AD:6A:01:13:86:5B:DF:1C:D4:10:2E:7D:07:59:AF:63:5A:7C:F4:72:0D:C9:63:C5:3
Alias name: globalsignrootcar6
  SHA1: 80:94:64:0E:B5:A7:A1:CA:11:9C:1F:DD:D5:9F:81:02:63:A7:FB:D1
  SHA256:
 2C:AB:EA:FE:37:D0:6C:A2:2A:BA:73:91:C0:03:3D:25:98:29:52:C4:53:64:73:49:76:3A:3A:B5:AD:6C:CF:6
Alias name: godaddyclass2ca
  SHA1: 27:96:BA:E6:3F:18:01:E2:77:26:1B:A0:D7:77:70:02:8F:20:EE:E4
  SHA256:
 C3:84:6B:F2:4B:9E:93:CA:64:27:4C:0E:C6:7C:1E:CC:5E:02:4F:FC:AC:D2:D7:40:19:35:0E:81:FE:54:6A:E
Alias name: godaddyrootcertificateauthorityg2
  SHA1: 47:BE:AB:C9:22:EA:E8:0E:78:78:34:62:A7:9F:45:C2:54:FD:E6:8B
  SHA256:
 45:14:0B:32:47:EB:9C:C8:C5:B4:F0:D7:B5:30:91:F7:32:92:08:9E:6E:5A:63:E2:74:9D:D3:AC:A9:19:8E:D
Alias name: godaddyrootg2ca
  SHA1: 47:BE:AB:C9:22:EA:E8:0E:78:78:34:62:A7:9F:45:C2:54:FD:E6:8B
  SHA256:
 45:14:0B:32:47:EB:9C:C8:C5:B4:F0:D7:B5:30:91:F7:32:92:08:9E:6E:5A:63:E2:74:9D:D3:AC:A9:19:8E:D
Alias name: gtsrootr1
  SHA1: E1:C9:50:E6:EF:22:F8:4C:56:45:72:8B:92:20:60:D7:D5:A7:A3:E8
  SHA256:
 2A:57:54:71:E3:13:40:BC:21:58:1C:BD:2C:F1:3E:15:84:63:20:3E:CE:94:BC:F9:D3:CC:19:6B:F0:9A:54:7
Alias name: gtsrootr2
  SHA1: D2:73:96:2A:2A:5E:39:9F:73:3F:E1:C7:1E:64:3F:03:38:34:FC:4D
  SHA256:
 C4:5D:7B:B0:8E:6D:67:E6:2E:42:35:11:0B:56:4E:5F:78:FD:92:EF:05:8C:84:0A:EA:4E:64:55:D7:58:5C:6
Alias name: gtsrootr3
  SHA1: 30:D4:24:6F:07:FF:DB:91:89:8A:0B:E9:49:66:11:EB:8C:5E:46:E5
  SHA256:
 15:D5:B8:77:46:19:EA:7D:54:CE:1C:A6:D0:B0:C4:03:E0:37:A9:17:F1:31:E8:A0:4E:1E:6B:7A:71:BA:BC:E
Alias name: gtsrootr4
  SHA1: 2A:1D:60:27:D9:4A:B1:0A:1C:4D:91:5C:CD:33:A0:CB:3E:2D:54:CB
  SHA256:
 71:CC:A5:39:1F:9E:79:4B:04:80:25:30:B3:63:E1:21:DA:8A:30:43:BB:26:66:2F:EA:4D:CA:7F:C9:51:A4:B
Alias name: hellenicacademicandresearchinstitutionseccrootca2015
```

```
  SHA1: 9F:F1:71:8D:92:D5:9A:F3:7D:74:97:B4:BC:6F:84:68:0B:BA:B6:66
  SHA256:
 44:B5:45:AA:8A:25:E6:5A:73:CA:15:DC:27:FC:36:D2:4C:1C:B9:95:3A:06:65:39:B1:15:82:DC:48:7B:48:3
Alias name: hellenicacademicandresearchinstitutionsrootca2011
  SHA1: FE:45:65:9B:79:03:5B:98:A1:61:B5:51:2E:AC:DA:58:09:48:22:4D
  SHA256:
 BC:10:4F:15:A4:8B:E7:09:DC:A5:42:A7:E1:D4:B9:DF:6F:05:45:27:E8:02:EA:A9:2D:59:54:44:25:8A:FE:7
Alias name: hellenicacademicandresearchinstitutionsrootca2015
  SHA1: 01:0C:06:95:A6:98:19:14:FF:BF:5F:C6:B0:B6:95:EA:29:E9:12:A6
  SHA256:
 A0:40:92:9A:02:CE:53:B4:AC:F4:F2:FF:C6:98:1C:E4:49:6F:75:5E:6D:45:FE:0B:2A:69:2B:CD:52:52:3F:3
Alias name: hongkongpostrootca1
  SHA1: D6:DA:A8:20:8D:09:D2:15:4D:24:B5:2F:CB:34:6E:B2:58:B2:8A:58
  SHA256:
 F9:E6:7D:33:6C:51:00:2A:C0:54:C6:32:02:2D:66:DD:A2:E7:E3:FF:F1:0A:D0:61:ED:31:D8:BB:B4:10:CF:B
Alias name: hongkongpostrootca3
  SHA1: 58:A2:D0:EC:20:52:81:5B:C1:F3:F8:64:02:24:4E:C2:8E:02:4B:02
  SHA256:
 5A:2F:C0:3F:0C:83:B0:90:BB:FA:40:60:4B:09:88:44:6C:76:36:18:3D:F9:84:6E:17:10:1A:44:7F:B8:EF:D
Alias name: identrustcommercialrootca1
  SHA1: DF:71:7E:AA:4A:D9:4E:C9:55:84:99:60:2D:48:DE:5F:BC:F0:3A:25
  SHA256:
 5D:56:49:9B:E4:D2:E0:8B:CF:CA:D0:8A:3E:38:72:3D:50:50:3B:DE:70:69:48:E4:2F:55:60:30:19:E5:28:A
Alias name: identrustpublicsectorrootca1
  SHA1: BA:29:41:60:77:98:3F:F4:F3:EF:F2:31:05:3B:2E:EA:6D:4D:45:FD
  SHA256:
 30:D0:89:5A:9A:44:8A:26:20:91:63:55:22:D1:F5:20:10:B5:86:7A:CA:E1:2C:78:EF:95:8F:D4:F4:38:9F:2
Alias name: isrgrootx1
  SHA1: CA:BD:2A:79:A1:07:6A:31:F2:1D:25:36:35:CB:03:9D:43:29:A5:E8
  SHA256:
 96:BC:EC:06:26:49:76:F3:74:60:77:9A:CF:28:C5:A7:CF:E8:A3:C0:AA:E1:1A:8F:FC:EE:05:C0:BD:DF:08:C
Alias name: izenpecom
  SHA1: 2F:78:3D:25:52:18:A7:4A:65:39:71:B5:2C:A2:9C:45:15:6F:E9:19
  SHA256:
 25:30:CC:8E:98:32:15:02:BA:D9:6F:9B:1F:BA:1B:09:9E:2D:29:9E:0F:45:48:BB:91:4F:36:3B:C0:D4:53:1
Alias name: keynectisrootca
  SHA1: 9C:61:5C:4D:4D:85:10:3A:53:26:C2:4D:BA:EA:E4:A2:D2:D5:CC:97
  SHA256:
 42:10:F1:99:49:9A:9A:C3:3C:8D:E0:2B:A6:DB:AA:14:40:8B:DD:8A:6E:32:46:89:C1:92:2D:06:97:15:A3:3
Alias name: microseceszignorootca2009
  SHA1: 89:DF:74:FE:5C:F4:0F:4A:80:F9:E3:37:7D:54:DA:91:E1:01:31:8E
  SHA256:
 3C:5F:81:FE:A5:FA:B8:2C:64:BF:A2:EA:EC:AF:CD:E8:E0:77:FC:86:20:A7:CA:E5:37:16:3D:F3:6E:DB:F3:7
Alias name: mozillacert0.pem
```

```
  SHA1: 97:81:79:50:D8:1C:96:70:CC:34:D8:09:CF:79:44:31:36:7E:F4:74
  SHA256:
 A5:31:25:18:8D:21:10:AA:96:4B:02:C7:B7:C6:DA:32:03:17:08:94:E5:FB:71:FF:FB:66:67:D5:E6:81:0A:3
Alias name: mozillacert1.pem
  SHA1: 23:E5:94:94:51:95:F2:41:48:03:B4:D5:64:D2:A3:A3:F5:D8:8B:8C
  SHA256:
 B4:41:0B:73:E2:E6:EA:CA:47:FB:C4:2F:8F:A4:01:8A:F4:38:1D:C5:4C:FA:A8:44:50:46:1E:ED:09:45:4D:E
Alias name: mozillacert10.pem
  SHA1: 5F:3A:FC:0A:8B:64:F6:86:67:34:74:DF:7E:A9:A2:FE:F9:FA:7A:51
  SHA256:
 21:DB:20:12:36:60:BB:2E:D4:18:20:5D:A1:1E:E7:A8:5A:65:E2:BC:6E:55:B5:AF:7E:78:99:C8:A2:66:D9:2
Alias name: mozillacert100.pem
  SHA1: 58:E8:AB:B0:36:15:33:FB:80:F7:9B:1B:6D:29:D3:FF:8D:5F:00:F0
  SHA256:
 49:E7:A4:42:AC:F0:EA:62:87:05:00:54:B5:25:64:B6:50:E4:F4:9E:42:E3:48:D6:AA:38:E0:39:E9:57:B1:C
Alias name: mozillacert101.pem
  SHA1: 99:A6:9B:E6:1A:FE:88:6B:4D:2B:82:00:7C:B8:54:FC:31:7E:15:39
  SHA256:
 62:F2:40:27:8C:56:4C:4D:D8:BF:7D:9D:4F:6F:36:6E:A8:94:D2:2F:5F:34:D9:89:A9:83:AC:EC:2F:FF:ED:5
Alias name: mozillacert102.pem
  SHA1: 96:C9:1B:0B:95:B4:10:98:42:FA:D0:D8:22:79:FE:60:FA:B9:16:83
  SHA256:
 EE:C5:49:6B:98:8C:E9:86:25:B9:34:09:2E:EC:29:08:BE:D0:B0:F3:16:C2:D4:73:0C:84:EA:F1:F3:D3:48:8
Alias name: mozillacert103.pem
  SHA1: 70:C1:8D:74:B4:28:81:0A:E4:FD:A5:75:D7:01:9F:99:B0:3D:50:74
  SHA256:
 3C:FC:3C:14:D1:F6:84:FF:17:E3:8C:43:CA:44:0C:00:B9:67:EC:93:3E:8B:FE:06:4C:A1:D7:2C:90:F2:AD:B
Alias name: mozillacert104.pem
  SHA1: 4F:99:AA:93:FB:2B:D1:37:26:A1:99:4A:CE:7F:F0:05:F2:93:5D:1E
  SHA256:
 1C:01:C6:F4:DB:B2:FE:FC:22:55:8B:2B:CA:32:56:3F:49:84:4A:CF:C3:2B:7B:E4:B0:FF:59:9F:9E:8C:7A:F
Alias name: mozillacert105.pem
  SHA1: 77:47:4F:C6:30:E4:0F:4C:47:64:3F:84:BA:B8:C6:95:4A:8A:41:EC
  SHA256:
 F0:9B:12:2C:71:14:F4:A0:9B:D4:EA:4F:4A:99:D5:58:B4:6E:4C:25:CD:81:14:0D:29:C0:56:13:91:4C:38:4
Alias name: mozillacert106.pem
  SHA1: E7:A1:90:29:D3:D5:52:DC:0D:0F:C6:92:D3:EA:88:0D:15:2E:1A:6B
  SHA256:
 D9:5F:EA:3C:A4:EE:DC:E7:4C:D7:6E:75:FC:6D:1F:F6:2C:44:1F:0F:A8:BC:77:F0:34:B1:9E:5D:B2:58:01:5
Alias name: mozillacert107.pem
  SHA1: 8E:1C:74:F8:A6:20:B9:E5:8A:F4:61:FA:EC:2B:47:56:51:1A:52:C6
  SHA256:
 F9:6F:23:F4:C3:E7:9C:07:7A:46:98:8D:5A:F5:90:06:76:A0:F0:39:CB:64:5D:D1:75:49:B2:16:C8:24:40:C
Alias name: mozillacert108.pem
```

```
  SHA1: B1:BC:96:8B:D4:F4:9D:62:2A:A8:9A:81:F2:15:01:52:A4:1D:82:9C
  SHA256:
 EB:D4:10:40:E4:BB:3E:C7:42:C9:E3:81:D3:1E:F2:A4:1A:48:B6:68:5C:96:E7:CE:F3:C1:DF:6C:D4:33:1C:9
Alias name: mozillacert109.pem
  SHA1: B5:61:EB:EA:A4:DE:E4:25:4B:69:1A:98:A5:57:47:C2:34:C7:D9:71
  SHA256:
 E2:3D:4A:03:6D:7B:70:E9:F5:95:B1:42:20:79:D2:B9:1E:DF:BB:1F:B6:51:A0:63:3E:AA:8A:9D:C5:F8:07:0
Alias name: mozillacert11.pem
  SHA1: 05:63:B8:63:0D:62:D7:5A:BB:C8:AB:1E:4B:DF:B5:A8:99:B2:4D:43
  SHA256:
 3E:90:99:B5:01:5E:8F:48:6C:00:BC:EA:9D:11:1E:E7:21:FA:BA:35:5A:89:BC:F1:DF:69:56:1E:3D:C6:32:5
Alias name: mozillacert110.pem
  SHA1: 93:05:7A:88:15:C6:4F:CE:88:2F:FA:91:16:52:28:78:BC:53:64:17
  SHA256:
 9A:6E:C0:12:E1:A7:DA:9D:BE:34:19:4D:47:8A:D7:C0:DB:18:22:FB:07:1D:F1:29:81:49:6E:D1:04:38:41:1
Alias name: mozillacert111.pem
  SHA1: 9C:BB:48:53:F6:A4:F6:D3:52:A4:E8:32:52:55:60:13:F5:AD:AF:65
  SHA256:
 59:76:90:07:F7:68:5D:0F:CD:50:87:2F:9F:95:D5:75:5A:5B:2B:45:7D:81:F3:69:2B:61:0A:98:67:2F:0E:1
Alias name: mozillacert112.pem
  SHA1: 43:13:BB:96:F1:D5:86:9B:C1:4E:6A:92:F6:CF:F6:34:69:87:82:37
  SHA256:
 DD:69:36:FE:21:F8:F0:77:C1:23:A1:A5:21:C1:22:24:F7:22:55:B7:3E:03:A7:26:06:93:E8:A2:4B:0F:A3:8
Alias name: mozillacert113.pem
  SHA1: 50:30:06:09:1D:97:D4:F5:AE:39:F7:CB:E7:92:7D:7D:65:2D:34:31
  SHA256:
 6D:C4:71:72:E0:1C:BC:B0:BF:62:58:0D:89:5F:E2:B8:AC:9A:D4:F8:73:80:1E:0C:10:B9:C8:37:D2:1E:B1:7
Alias name: mozillacert114.pem
  SHA1: 51:C6:E7:08:49:06:6E:F3:92:D4:5C:A0:0D:6D:A3:62:8F:C3:52:39
  SHA256:
 B0:BF:D5:2B:B0:D7:D9:BD:92:BF:5D:4D:C1:3D:A2:55:C0:2C:54:2F:37:83:65:EA:89:39:11:F5:5E:55:F2:3
Alias name: mozillacert115.pem
  SHA1: 59:0D:2D:7D:88:4F:40:2E:61:7E:A5:62:32:17:65:CF:17:D8:94:E9
  SHA256:
 91:E2:F5:78:8D:58:10:EB:A7:BA:58:73:7D:E1:54:8A:8E:CA:CD:01:45:98:BC:0B:14:3E:04:1B:17:05:25:5
Alias name: mozillacert116.pem
  SHA1: 2B:B1:F5:3E:55:0C:1D:C5:F1:D4:E6:B7:6A:46:4B:55:06:02:AC:21
  SHA256:
 F3:56:BE:A2:44:B7:A9:1E:B3:5D:53:CA:9A:D7:86:4A:CE:01:8E:2D:35:D5:F8:F9:6D:DF:68:A6:F4:1A:A4:7
Alias name: mozillacert117.pem
  SHA1: D4:DE:20:D0:5E:66:FC:53:FE:1A:50:88:2C:78:DB:28:52:CA:E4:74
  SHA256:
 16:AF:57:A9:F6:76:B0:AB:12:60:95:AA:5E:BA:DE:F2:2A:B3:11:19:D6:44:AC:95:CD:4B:93:DB:F3:F2:6A:E
Alias name: mozillacert118.pem
```

```
  SHA1: 7E:78:4A:10:1C:82:65:CC:2D:E1:F1:6D:47:B4:40:CA:D9:0A:19:45
  SHA256:
 5F:0B:62:EA:B5:E3:53:EA:65:21:65:16:58:FB:B6:53:59:F4:43:28:0A:4A:FB:D1:04:D7:7D:10:F9:F0:4C:0
Alias name: mozillacert119.pem
  SHA1: 75:E0:AB:B6:13:85:12:27:1C:04:F8:5F:DD:DE:38:E4:B7:24:2E:FE
  SHA256:
 CA:42:DD:41:74:5F:D0:B8:1E:B9:02:36:2C:F9:D8:BF:71:9D:A1:BD:1B:1E:FC:94:6F:5B:4C:99:F4:2C:1B:9
Alias name: mozillacert12.pem
  SHA1: A8:98:5D:3A:65:E5:E5:C4:B2:D7:D6:6D:40:C6:DD:2F:B1:9C:54:36
  SHA256:
 43:48:A0:E9:44:4C:78:CB:26:5E:05:8D:5E:89:44:B4:D8:4F:96:62:BD:26:DB:25:7F:89:34:A4:43:C7:01:6
Alias name: mozillacert120.pem
  SHA1: DA:40:18:8B:91:89:A3:ED:EE:AE:DA:97:FE:2F:9D:F5:B7:D1:8A:41
  SHA256:
 CF:56:FF:46:A4:A1:86:10:9D:D9:65:84:B5:EE:B5:8A:51:0C:42:75:B0:E5:F9:4F:40:BB:AE:86:5E:19:F6:7
Alias name: mozillacert121.pem
  SHA1: CC:AB:0E:A0:4C:23:01:D6:69:7B:DD:37:9F:CD:12:EB:24:E3:94:9D
  SHA256:
 8C:72:09:27:9A:C0:4E:27:5E:16:D0:7F:D3:B7:75:E8:01:54:B5:96:80:46:E3:1F:52:DD:25:76:63:24:E9:A
Alias name: mozillacert122.pem
  SHA1: 02:FA:F3:E2:91:43:54:68:60:78:57:69:4D:F5:E4:5B:68:85:18:68
  SHA256:
 68:7F:A4:51:38:22:78:FF:F0:C8:B1:1F:8D:43:D5:76:67:1C:6E:B2:BC:EA:B4:13:FB:83:D9:65:D0:6D:2F:F
Alias name: mozillacert123.pem
  SHA1: 2A:B6:28:48:5E:78:FB:F3:AD:9E:79:10:DD:6B:DF:99:72:2C:96:E5
  SHA256:
 07:91:CA:07:49:B2:07:82:AA:D3:C7:D7:BD:0C:DF:C9:48:58:35:84:3E:B2:D7:99:60:09:CE:43:AB:6C:69:2
Alias name: mozillacert124.pem
  SHA1: 4D:23:78:EC:91:95:39:B5:00:7F:75:8F:03:3B:21:1E:C5:4D:8B:CF
  SHA256:
 80:95:21:08:05:DB:4B:BC:35:5E:44:28:D8:FD:6E:C2:CD:E3:AB:5F:B9:7A:99:42:98:8E:B8:F4:DC:D0:60:1
Alias name: mozillacert125.pem
  SHA1: B3:1E:B1:B7:40:E3:6C:84:02:DA:DC:37:D4:4D:F5:D4:67:49:52:F9
  SHA256:
 73:C1:76:43:4F:1B:C6:D5:AD:F4:5B:0E:76:E7:27:28:7C:8D:E5:76:16:C1:E6:E6:14:1A:2B:2C:BC:7D:8E:4
Alias name: mozillacert126.pem
  SHA1: 25:01:90:19:CF:FB:D9:99:1C:B7:68:25:74:8D:94:5F:30:93:95:42
  SHA256:
 AF:8B:67:62:A1:E5:28:22:81:61:A9:5D:5C:55:9E:E2:66:27:8F:75:D7:9E:83:01:89:A5:03:50:6A:BD:6B:4
Alias name: mozillacert127.pem
  SHA1: DE:28:F4:A4:FF:E5:B9:2F:A3:C5:03:D1:A3:49:A7:F9:96:2A:82:12
  SHA256:
 FF:85:6A:2D:25:1D:CD:88:D3:66:56:F4:50:12:67:98:CF:AB:AA:DE:40:79:9C:72:2D:E4:D2:B5:DB:36:A7:3
Alias name: mozillacert128.pem
```

```
  SHA1: A9:E9:78:08:14:37:58:88:F2:05:19:B0:6D:2B:0D:2B:60:16:90:7D
  SHA256:
 CA:2D:82:A0:86:77:07:2F:8A:B6:76:4F:F0:35:67:6C:FE:3E:5E:32:5E:01:21:72:DF:3F:92:09:6D:B7:9B:8
Alias name: mozillacert129.pem
  SHA1: E6:21:F3:35:43:79:05:9A:4B:68:30:9D:8A:2F:74:22:15:87:EC:79
  SHA256:
 A0:45:9B:9F:63:B2:25:59:F5:FA:5D:4C:6D:B3:F9:F7:2F:F1:93:42:03:35:78:F0:73:BF:1D:1B:46:CB:B9:1
Alias name: mozillacert13.pem
  SHA1: 06:08:3F:59:3F:15:A1:04:A0:69:A4:6B:A9:03:D0:06:B7:97:09:91
  SHA256:
 6C:61:DA:C3:A2:DE:F0:31:50:6B:E0:36:D2:A6:FE:40:19:94:FB:D1:3D:F9:C8:D4:66:59:92:74:C4:46:EC:9
Alias name: mozillacert130.pem
  SHA1: E5:DF:74:3C:B6:01:C4:9B:98:43:DC:AB:8C:E8:6A:81:10:9F:E4:8E
  SHA256:
 F4:C1:49:55:1A:30:13:A3:5B:C7:BF:FE:17:A7:F3:44:9B:C1:AB:5B:5A:0A:E7:4B:06:C2:3B:90:00:4C:01:0
Alias name: mozillacert131.pem
  SHA1: 37:9A:19:7B:41:85:45:35:0C:A6:03:69:F3:3C:2E:AF:47:4F:20:79
  SHA256:
 A0:23:4F:3B:C8:52:7C:A5:62:8E:EC:81:AD:5D:69:89:5D:A5:68:0D:C9:1D:1C:B8:47:7F:33:F8:78:B9:5B:0
Alias name: mozillacert132.pem
  SHA1: 39:21:C1:15:C1:5D:0E:CA:5C:CB:5B:C4:F0:7D:21:D8:05:0B:56:6A
  SHA256:
 77:40:73:12:C6:3A:15:3D:5B:C0:0B:4E:51:75:9C:DF:DA:C2:37:DC:2A:33:B6:79:46:E9:8E:9B:FA:68:0A:E
Alias name: mozillacert133.pem
  SHA1: 85:B5:FF:67:9B:0C:79:96:1F:C8:6E:44:22:00:46:13:DB:17:92:84
  SHA256:
 7D:3B:46:5A:60:14:E5:26:C0:AF:FC:EE:21:27:D2:31:17:27:AD:81:1C:26:84:2D:00:6A:F3:73:06:CC:80:B
Alias name: mozillacert134.pem
  SHA1: 70:17:9B:86:8C:00:A4:FA:60:91:52:22:3F:9F:3E:32:BD:E0:05:62
  SHA256:
 69:FA:C9:BD:55:FB:0A:C7:8D:53:BB:EE:5C:F1:D5:97:98:9F:D0:AA:AB:20:A2:51:51:BD:F1:73:3E:E7:D1:2
Alias name: mozillacert135.pem
  SHA1: 62:52:DC:40:F7:11:43:A2:2F:DE:9E:F7:34:8E:06:42:51:B1:81:18
  SHA256:
 D8:E0:FE:BC:1D:B2:E3:8D:00:94:0F:37:D2:7D:41:34:4D:99:3E:73:4B:99:D5:65:6D:97:78:D4:D8:14:36:2
Alias name: mozillacert136.pem
  SHA1: D1:EB:23:A4:6D:17:D6:8F:D9:25:64:C2:F1:F1:60:17:64:D8:E3:49
  SHA256:
 D7:A7:A0:FB:5D:7E:27:31:D7:71:E9:48:4E:BC:DE:F7:1D:5F:0C:3E:0A:29:48:78:2B:C8:3E:E0:EA:69:9E:F
Alias name: mozillacert137.pem
  SHA1: 4A:65:D5:F4:1D:EF:39:B8:B8:90:4A:4A:D3:64:81:33:CF:C7:A1:D1
  SHA256:
 BD:81:CE:3B:4F:65:91:D1:1A:67:B5:FC:7A:47:FD:EF:25:52:1B:F9:AA:4E:18:B9:E3:DF:2E:34:A7:80:3B:E
Alias name: mozillacert138.pem
```

```
  SHA1: E1:9F:E3:0E:8B:84:60:9E:80:9B:17:0D:72:A8:C5:BA:6E:14:09:BD
  SHA256:
 3F:06:E5:56:81:D4:96:F5:BE:16:9E:B5:38:9F:9F:2B:8F:F6:1E:17:08:DF:68:81:72:48:49:CD:5D:27:CB:6
Alias name: mozillacert139.pem
  SHA1: DE:3F:40:BD:50:93:D3:9B:6C:60:F6:DA:BC:07:62:01:00:89:76:C9
  SHA256:
 A4:5E:DE:3B:BB:F0:9C:8A:E1:5C:72:EF:C0:72:68:D6:93:A2:1C:99:6F:D5:1E:67:CA:07:94:60:FD:6D:88:7
Alias name: mozillacert14.pem
  SHA1: 5F:B7:EE:06:33:E2:59:DB:AD:0C:4C:9A:E6:D3:8F:1A:61:C7:DC:25
  SHA256:
 74:31:E5:F4:C3:C1:CE:46:90:77:4F:0B:61:E0:54:40:88:3B:A9:A0:1E:D0:0B:A6:AB:D7:80:6E:D3:B1:18:C
Alias name: mozillacert140.pem
  SHA1: CA:3A:FB:CF:12:40:36:4B:44:B2:16:20:88:80:48:39:19:93:7C:F7
  SHA256:
 85:A0:DD:7D:D7:20:AD:B7:FF:05:F8:3D:54:2B:20:9D:C7:FF:45:28:F7:D6:77:B1:83:89:FE:A5:E5:C4:9E:8
Alias name: mozillacert141.pem
  SHA1: 31:7A:2A:D0:7F:2B:33:5E:F5:A1:C3:4E:4B:57:E8:B7:D8:F1:FC:A6
  SHA256:
 58:D0:17:27:9C:D4:DC:63:AB:DD:B1:96:A6:C9:90:6C:30:C4:E0:87:83:EA:E8:C1:60:99:54:D6:93:55:59:6
Alias name: mozillacert142.pem
  SHA1: 1F:49:14:F7:D8:74:95:1D:DD:AE:02:C0:BE:FD:3A:2D:82:75:51:85
  SHA256:
 18:F1:FC:7F:20:5D:F8:AD:DD:EB:7F:E0:07:DD:57:E3:AF:37:5A:9C:4D:8D:73:54:6B:F4:F1:FE:D1:E1:8D:3
Alias name: mozillacert143.pem
  SHA1: 36:B1:2B:49:F9:81:9E:D7:4C:9E:BC:38:0F:C6:56:8F:5D:AC:B2:F7
  SHA256:
 E7:5E:72:ED:9F:56:0E:EC:6E:B4:80:00:73:A4:3F:C3:AD:19:19:5A:39:22:82:01:78:95:97:4A:99:02:6B:6
Alias name: mozillacert144.pem
  SHA1: 37:F7:6D:E6:07:7C:90:C5:B1:3E:93:1A:B7:41:10:B4:F2:E4:9A:27
  SHA256:
 79:08:B4:03:14:C1:38:10:0B:51:8D:07:35:80:7F:FB:FC:F8:51:8A:00:95:33:71:05:BA:38:6B:15:3D:D9:2
Alias name: mozillacert145.pem
  SHA1: 10:1D:FA:3F:D5:0B:CB:BB:9B:B5:60:0C:19:55:A4:1A:F4:73:3A:04
  SHA256:
 D4:1D:82:9E:8C:16:59:82:2A:F9:3F:CE:62:BF:FC:DE:26:4F:C8:4E:8B:95:0C:5F:F2:75:D0:52:35:46:95:A
Alias name: mozillacert146.pem
  SHA1: 21:FC:BD:8E:7F:6C:AF:05:1B:D1:B3:43:EC:A8:E7:61:47:F2:0F:8A
  SHA256:
 48:98:C6:88:8C:0C:FF:B0:D3:E3:1A:CA:8A:37:D4:E3:51:5F:F7:46:D0:26:35:D8:66:46:CF:A0:A3:18:5A:E
Alias name: mozillacert147.pem
  SHA1: 58:11:9F:0E:12:82:87:EA:50:FD:D9:87:45:6F:4F:78:DC:FA:D6:D4
  SHA256:
 85:FB:2F:91:DD:12:27:5A:01:45:B6:36:53:4F:84:02:4A:D6:8B:69:B8:EE:88:68:4F:F7:11:37:58:05:B3:4
Alias name: mozillacert148.pem
```

```
  SHA1: 04:83:ED:33:99:AC:36:08:05:87:22:ED:BC:5E:46:00:E3:BE:F9:D7
  SHA256:
 6E:A5:47:41:D0:04:66:7E:ED:1B:48:16:63:4A:A3:A7:9E:6E:4B:96:95:0F:82:79:DA:FC:8D:9B:D8:81:21:3
Alias name: mozillacert149.pem
  SHA1: 6E:3A:55:A4:19:0C:19:5C:93:84:3C:C0:DB:72:2E:31:30:61:F0:B1
  SHA256:
 0C:25:8A:12:A5:67:4A:EF:25:F2:8B:A7:DC:FA:EC:EE:A3:48:E5:41:E6:F5:CC:4E:E6:3B:71:B3:61:60:6A:C
Alias name: mozillacert15.pem
  SHA1: 74:20:74:41:72:9C:DD:92:EC:79:31:D8:23:10:8D:C2:81:92:E2:BB
  SHA256:
 0F:99:3C:8A:EF:97:BA:AF:56:87:14:0E:D5:9A:D1:82:1B:B4:AF:AC:F0:AA:9A:58:B5:D5:7A:33:8A:3A:FB:C
Alias name: mozillacert150.pem
  SHA1: 33:9B:6B:14:50:24:9B:55:7A:01:87:72:84:D9:E0:2F:C3:D2:D8:E9
  SHA256:
 EF:3C:B4:17:FC:8E:BF:6F:97:87:6C:9E:4E:CE:39:DE:1E:A5:FE:64:91:41:D1:02:8B:7D:11:C0:B2:29:8C:E
Alias name: mozillacert151.pem
  SHA1: AC:ED:5F:65:53:FD:25:CE:01:5F:1F:7A:48:3B:6A:74:9F:61:78:C6
  SHA256:
 7F:12:CD:5F:7E:5E:29:0E:C7:D8:51:79:D5:B7:2C:20:A5:BE:75:08:FF:DB:5B:F8:1A:B9:68:4A:7F:C9:F6:6
Alias name: mozillacert16.pem
  SHA1: DA:C9:02:4F:54:D8:F6:DF:94:93:5F:B1:73:26:38:CA:6A:D7:7C:13
  SHA256:
 06:87:26:03:31:A7:24:03:D9:09:F1:05:E6:9B:CF:0D:32:E1:BD:24:93:FF:C6:D9:20:6D:11:BC:D6:77:07:3
Alias name: mozillacert17.pem
  SHA1: 40:54:DA:6F:1C:3F:40:74:AC:ED:0F:EC:CD:DB:79:D1:53:FB:90:1D
  SHA256:
 76:7C:95:5A:76:41:2C:89:AF:68:8E:90:A1:C7:0F:55:6C:FD:6B:60:25:DB:EA:10:41:6D:7E:B6:83:1F:8C:4
Alias name: mozillacert18.pem
  SHA1: 79:98:A3:08:E1:4D:65:85:E6:C2:1E:15:3A:71:9F:BA:5A:D3:4A:D9
  SHA256:
 44:04:E3:3B:5E:14:0D:CF:99:80:51:FD:FC:80:28:C7:C8:16:15:C5:EE:73:7B:11:1B:58:82:33:A9:B5:35:A
Alias name: mozillacert19.pem
  SHA1: B4:35:D4:E1:11:9D:1C:66:90:A7:49:EB:B3:94:BD:63:7B:A7:82:B7
  SHA256:
 C4:70:CF:54:7E:23:02:B9:77:FB:29:DD:71:A8:9A:7B:6C:1F:60:77:7B:03:29:F5:60:17:F3:28:BF:4F:6B:E
Alias name: mozillacert2.pem
  SHA1: 22:D5:D8:DF:8F:02:31:D1:8D:F7:9D:B7:CF:8A:2D:64:C9:3F:6C:3A
  SHA256:
 69:DD:D7:EA:90:BB:57:C9:3E:13:5D:C8:5E:A6:FC:D5:48:0B:60:32:39:BD:C4:54:FC:75:8B:2A:26:CF:7F:7
Alias name: mozillacert20.pem
  SHA1: D8:C5:38:8A:B7:30:1B:1B:6E:D4:7A:E6:45:25:3A:6F:9F:1A:27:61
  SHA256:
 62:DD:0B:E9:B9:F5:0A:16:3E:A0:F8:E7:5C:05:3B:1E:CA:57:EA:55:C8:68:8F:64:7C:68:81:F2:C8:35:7B:9
Alias name: mozillacert21.pem
```

```
  SHA1: 9B:AA:E5:9F:56:EE:21:CB:43:5A:BE:25:93:DF:A7:F0:40:D1:1D:CB
  SHA256:
 BE:6C:4D:A2:BB:B9:BA:59:B6:F3:93:97:68:37:42:46:C3:C0:05:99:3F:A9:8F:02:0D:1D:ED:BE:D4:8A:81:D
Alias name: mozillacert22.pem
  SHA1: 32:3C:11:8E:1B:F7:B8:B6:52:54:E2:E2:10:0D:D6:02:90:37:F0:96
  SHA256:
 37:D5:10:06:C5:12:EA:AB:62:64:21:F1:EC:8C:92:01:3F:C5:F8:2A:E9:8E:E5:33:EB:46:19:B8:DE:B4:D0:6
Alias name: mozillacert23.pem
  SHA1: 91:C6:D6:EE:3E:8A:C8:63:84:E5:48:C2:99:29:5C:75:6C:81:7B:81
  SHA256:
 8D:72:2F:81:A9:C1:13:C0:79:1D:F1:36:A2:96:6D:B2:6C:95:0A:97:1D:B4:6B:41:99:F4:EA:54:B7:8B:FB:9
Alias name: mozillacert24.pem
  SHA1: 59:AF:82:79:91:86:C7:B4:75:07:CB:CF:03:57:46:EB:04:DD:B7:16
  SHA256:
 66:8C:83:94:7D:A6:3B:72:4B:EC:E1:74:3C:31:A0:E6:AE:D0:DB:8E:C5:B3:1B:E3:77:BB:78:4F:91:B6:71:6
Alias name: mozillacert25.pem
  SHA1: 4E:B6:D5:78:49:9B:1C:CF:5F:58:1E:AD:56:BE:3D:9B:67:44:A5:E5
  SHA256:
 9A:CF:AB:7E:43:C8:D8:80:D0:6B:26:2A:94:DE:EE:E4:B4:65:99:89:C3:D0:CA:F1:9B:AF:64:05:E4:1A:B7:D
Alias name: mozillacert26.pem
  SHA1: 87:82:C6:C3:04:35:3B:CF:D2:96:92:D2:59:3E:7D:44:D9:34:FF:11
  SHA256:
 F1:C1:B5:0A:E5:A2:0D:D8:03:0E:C9:F6:BC:24:82:3D:D3:67:B5:25:57:59:B4:E7:1B:61:FC:E9:F7:37:5D:7
Alias name: mozillacert27.pem
  SHA1: 3A:44:73:5A:E5:81:90:1F:24:86:61:46:1E:3B:9C:C4:5F:F5:3A:1B
  SHA256:
 42:00:F5:04:3A:C8:59:0E:BB:52:7D:20:9E:D1:50:30:29:FB:CB:D4:1C:A1:B5:06:EC:27:F1:5A:DE:7D:AC:6
Alias name: mozillacert28.pem
  SHA1: 66:31:BF:9E:F7:4F:9E:B6:C9:D5:A6:0C:BA:6A:BE:D1:F7:BD:EF:7B
  SHA256:
 0C:2C:D6:3D:F7:80:6F:A3:99:ED:E8:09:11:6B:57:5B:F8:79:89:F0:65:18:F9:80:8C:86:05:03:17:8B:AF:6
Alias name: mozillacert29.pem
  SHA1: 74:F8:A3:C3:EF:E7:B3:90:06:4B:83:90:3C:21:64:60:20:E5:DF:CE
  SHA256:
 15:F0:BA:00:A3:AC:7A:F3:AC:88:4C:07:2B:10:11:A0:77:BD:77:C0:97:F4:01:64:B2:F8:59:8A:BD:83:86:0
Alias name: mozillacert3.pem
  SHA1: 87:9F:4B:EE:05:DF:98:58:3B:E3:60:D6:33:E7:0D:3F:FE:98:71:AF
  SHA256:
 39:DF:7B:68:2B:7B:93:8F:84:71:54:81:CC:DE:8D:60:D8:F2:2E:C5:98:87:7D:0A:AA:C1:2B:59:18:2B:03:1
Alias name: mozillacert30.pem
  SHA1: E7:B4:F6:9D:61:EC:90:69:DB:7E:90:A7:40:1A:3C:F4:7D:4F:E8:EE
  SHA256:
 A7:12:72:AE:AA:A3:CF:E8:72:7F:7F:B3:9F:0F:B3:D1:E5:42:6E:90:60:B0:6E:E6:F1:3E:9A:3C:58:33:CD:4
Alias name: mozillacert31.pem
```

```
  SHA1: 9F:74:4E:9F:2B:4D:BA:EC:0F:31:2C:50:B6:56:3B:8E:2D:93:C3:11
  SHA256:
 17:93:92:7A:06:14:54:97:89:AD:CE:2F:8F:34:F7:F0:B6:6D:0F:3A:E3:A3:B8:4D:21:EC:15:DB:BA:4F:AD:C
Alias name: mozillacert32.pem
  SHA1: 60:D6:89:74:B5:C2:65:9E:8A:0F:C1:88:7C:88:D2:46:69:1B:18:2C
  SHA256:
 B9:BE:A7:86:0A:96:2E:A3:61:1D:AB:97:AB:6D:A3:E2:1C:10:68:B9:7D:55:57:5E:D0:E1:12:79:C1:1C:89:3
Alias name: mozillacert33.pem
  SHA1: FE:B8:C4:32:DC:F9:76:9A:CE:AE:3D:D8:90:8F:FD:28:86:65:64:7D
  SHA256:
 A2:2D:BA:68:1E:97:37:6E:2D:39:7D:72:8A:AE:3A:9B:62:96:B9:FD:BA:60:BC:2E:11:F6:47:F2:C6:75:FB:3
Alias name: mozillacert34.pem
  SHA1: 59:22:A1:E1:5A:EA:16:35:21:F8:98:39:6A:46:46:B0:44:1B:0F:A9
  SHA256:
 41:C9:23:86:6A:B4:CA:D6:B7:AD:57:80:81:58:2E:02:07:97:A6:CB:DF:4F:FF:78:CE:83:96:B3:89:37:D7:F
Alias name: mozillacert35.pem
  SHA1: 2A:C8:D5:8B:57:CE:BF:2F:49:AF:F2:FC:76:8F:51:14:62:90:7A:41
  SHA256:
 92:BF:51:19:AB:EC:CA:D0:B1:33:2D:C4:E1:D0:5F:BA:75:B5:67:90:44:EE:0C:A2:6E:93:1F:74:4F:2F:33:C
Alias name: mozillacert36.pem
  SHA1: 23:88:C9:D3:71:CC:9E:96:3D:FF:7D:3C:A7:CE:FC:D6:25:EC:19:0D
  SHA256:
 32:7A:3D:76:1A:BA:DE:A0:34:EB:99:84:06:27:5C:B1:A4:77:6E:FD:AE:2F:DF:6D:01:68:EA:1C:4F:55:67:D
Alias name: mozillacert37.pem
  SHA1: B1:2E:13:63:45:86:A4:6F:1A:B2:60:68:37:58:2D:C4:AC:FD:94:97
  SHA256:
 E3:B6:A2:DB:2E:D7:CE:48:84:2F:7A:C5:32:41:C7:B7:1D:54:14:4B:FB:40:C1:1F:3F:1D:0B:42:F5:EE:A1:2
Alias name: mozillacert38.pem
  SHA1: CB:A1:C5:F8:B0:E3:5E:B8:B9:45:12:D3:F9:34:A2:E9:06:10:D3:36
  SHA256:
 A6:C5:1E:0D:A5:CA:0A:93:09:D2:E4:C0:E4:0C:2A:F9:10:7A:AE:82:03:85:7F:E1:98:E3:E7:69:E3:43:08:5
Alias name: mozillacert39.pem
  SHA1: AE:50:83:ED:7C:F4:5C:BC:8F:61:C6:21:FE:68:5D:79:42:21:15:6E
  SHA256:
 E6:B8:F8:76:64:85:F8:07:AE:7F:8D:AC:16:70:46:1F:07:C0:A1:3E:EF:3A:1F:F7:17:53:8D:7A:BA:D3:91:B
Alias name: mozillacert4.pem
  SHA1: E3:92:51:2F:0A:CF:F5:05:DF:F6:DE:06:7F:75:37:E1:65:EA:57:4B
  SHA256:
 0B:5E:ED:4E:84:64:03:CF:55:E0:65:84:84:40:ED:2A:82:75:8B:F5:B9:AA:1F:25:3D:46:13:CF:A0:80:FF:3
Alias name: mozillacert40.pem
  SHA1: 80:25:EF:F4:6E:70:C8:D4:72:24:65:84:FE:40:3B:8A:8D:6A:DB:F5
  SHA256:
 8D:A0:84:FC:F9:9C:E0:77:22:F8:9B:32:05:93:98:06:FA:5C:B8:11:E1:C8:13:F6:A1:08:C7:D3:36:B3:40:8
Alias name: mozillacert41.pem
```

```
  SHA1: 6B:2F:34:AD:89:58:BE:62:FD:B0:6B:5C:CE:BB:9D:D9:4F:4E:39:F3
  SHA256:
 EB:F3:C0:2A:87:89:B1:FB:7D:51:19:95:D6:63:B7:29:06:D9:13:CE:0D:5E:10:56:8A:8A:77:E2:58:61:67:E
Alias name: mozillacert42.pem
  SHA1: 85:A4:08:C0:9C:19:3E:5D:51:58:7D:CD:D6:13:30:FD:8C:DE:37:BF
  SHA256:
 B6:19:1A:50:D0:C3:97:7F:7D:A9:9B:CD:AA:C8:6A:22:7D:AE:B9:67:9E:C7:0B:A3:B0:C9:D9:22:71:C1:70:D
Alias name: mozillacert43.pem
  SHA1: F9:CD:0E:2C:DA:76:24:C1:8F:BD:F0:F0:AB:B6:45:B8:F7:FE:D5:7A
  SHA256:
 50:79:41:C7:44:60:A0:B4:70:86:22:0D:4E:99:32:57:2A:B5:D1:B5:BB:CB:89:80:AB:1C:B1:76:51:A8:44:D
Alias name: mozillacert44.pem
  SHA1: 5F:43:E5:B1:BF:F8:78:8C:AC:1C:C7:CA:4A:9A:C6:22:2B:CC:34:C6
  SHA256:
 96:0A:DF:00:63:E9:63:56:75:0C:29:65:DD:0A:08:67:DA:0B:9C:BD:6E:77:71:4A:EA:FB:23:49:AB:39:3D:A
Alias name: mozillacert45.pem
  SHA1: 67:65:0D:F1:7E:8E:7E:5B:82:40:A4:F4:56:4B:CF:E2:3D:69:C6:F0
  SHA256:
 C0:A6:F4:DC:63:A2:4B:FD:CF:54:EF:2A:6A:08:2A:0A:72:DE:35:80:3E:2F:F5:FF:52:7A:E5:D8:72:06:DF:D
Alias name: mozillacert46.pem
  SHA1: 40:9D:4B:D9:17:B5:5C:27:B6:9B:64:CB:98:22:44:0D:CD:09:B8:89
  SHA256:
 EC:C3:E9:C3:40:75:03:BE:E0:91:AA:95:2F:41:34:8F:F8:8B:AA:86:3B:22:64:BE:FA:C8:07:90:15:74:E9:3
Alias name: mozillacert47.pem
  SHA1: 1B:4B:39:61:26:27:6B:64:91:A2:68:6D:D7:02:43:21:2D:1F:1D:96
  SHA256:
 E4:C7:34:30:D7:A5:B5:09:25:DF:43:37:0A:0D:21:6E:9A:79:B9:D6:DB:83:73:A0:C6:9E:B1:CC:31:C7:C5:2
Alias name: mozillacert48.pem
  SHA1: A0:A1:AB:90:C9:FC:84:7B:3B:12:61:E8:97:7D:5F:D3:22:61:D3:CC
  SHA256:
 0F:4E:9C:DD:26:4B:02:55:50:D1:70:80:63:40:21:4F:E9:44:34:C9:B0:2F:69:7E:C7:10:FC:5F:EA:FB:5E:3
Alias name: mozillacert49.pem
  SHA1: 61:57:3A:11:DF:0E:D8:7E:D5:92:65:22:EA:D0:56:D7:44:B3:23:71
  SHA256:
 B7:B1:2B:17:1F:82:1D:AA:99:0C:D0:FE:50:87:B1:28:44:8B:A8:E5:18:4F:84:C5:1E:02:B5:C8:FB:96:2B:2
Alias name: mozillacert5.pem
  SHA1: B8:01:86:D1:EB:9C:86:A5:41:04:CF:30:54:F3:4C:52:B7:E5:58:C6
  SHA256:
 CE:CD:DC:90:50:99:D8:DA:DF:C5:B1:D2:09:B7:37:CB:E2:C1:8C:FB:2C:10:C0:FF:0B:CF:0D:32:86:FC:1A:A
Alias name: mozillacert50.pem
  SHA1: 8C:96:BA:EB:DD:2B:07:07:48:EE:30:32:66:A0:F3:98:6E:7C:AE:58
  SHA256:
 35:AE:5B:DD:D8:F7:AE:63:5C:FF:BA:56:82:A8:F0:0B:95:F4:84:62:C7:10:8E:E9:A0:E5:29:2B:07:4A:AF:B
Alias name: mozillacert51.pem
```

```
  SHA1: FA:B7:EE:36:97:26:62:FB:2D:B0:2A:F6:BF:03:FD:E8:7C:4B:2F:9B
  SHA256:
 EA:A9:62:C4:FA:4A:6B:AF:EB:E4:15:19:6D:35:1C:CD:88:8D:4F:53:F3:FA:8A:E6:D7:C4:66:A9:4E:60:42:B
Alias name: mozillacert52.pem
  SHA1: 8B:AF:4C:9B:1D:F0:2A:92:F7:DA:12:8E:B9:1B:AC:F4:98:60:4B:6F
  SHA256:
 E2:83:93:77:3D:A8:45:A6:79:F2:08:0C:C7:FB:44:A3:B7:A1:C3:79:2C:B7:EB:77:29:FD:CB:6A:8D:99:AE:A
Alias name: mozillacert53.pem
  SHA1: 7F:8A:B0:CF:D0:51:87:6A:66:F3:36:0F:47:C8:8D:8C:D3:35:FC:74
  SHA256:
 2D:47:43:7D:E1:79:51:21:5A:12:F3:C5:8E:51:C7:29:A5:80:26:EF:1F:CC:0A:5F:B3:D9:DC:01:2F:60:0D:1
Alias name: mozillacert54.pem
  SHA1: 03:9E:ED:B8:0B:E7:A0:3C:69:53:89:3B:20:D2:D9:32:3A:4C:2A:FD
  SHA256:
 B4:78:B8:12:25:0D:F8:78:63:5C:2A:A7:EC:7D:15:5E:AA:62:5E:E8:29:16:E2:CD:29:43:61:88:6C:D1:FB:D
Alias name: mozillacert55.pem
  SHA1: AA:DB:BC:22:23:8F:C4:01:A1:27:BB:38:DD:F4:1D:DB:08:9E:F0:12
  SHA256:
 A4:31:0D:50:AF:18:A6:44:71:90:37:2A:86:AF:AF:8B:95:1F:FB:43:1D:83:7F:1E:56:88:B4:59:71:ED:15:5
Alias name: mozillacert56.pem
  SHA1: F1:8B:53:8D:1B:E9:03:B6:A6:F0:56:43:5B:17:15:89:CA:F3:6B:F2
  SHA256:
 4B:03:F4:58:07:AD:70:F2:1B:FC:2C:AE:71:C9:FD:E4:60:4C:06:4C:F5:FF:B6:86:BA:E5:DB:AA:D7:FD:D3:4
Alias name: mozillacert57.pem
  SHA1: D6:DA:A8:20:8D:09:D2:15:4D:24:B5:2F:CB:34:6E:B2:58:B2:8A:58
  SHA256:
 F9:E6:7D:33:6C:51:00:2A:C0:54:C6:32:02:2D:66:DD:A2:E7:E3:FF:F1:0A:D0:61:ED:31:D8:BB:B4:10:CF:B
Alias name: mozillacert58.pem
  SHA1: 8D:17:84:D5:37:F3:03:7D:EC:70:FE:57:8B:51:9A:99:E6:10:D7:B0
  SHA256:
 5E:DB:7A:C4:3B:82:A0:6A:87:61:E8:D7:BE:49:79:EB:F2:61:1F:7D:D7:9B:F9:1C:1C:6B:56:6A:21:9E:D7:6
Alias name: mozillacert59.pem
  SHA1: 36:79:CA:35:66:87:72:30:4D:30:A5:FB:87:3B:0F:A7:7B:B7:0D:54
  SHA256:
 23:99:56:11:27:A5:71:25:DE:8C:EF:EA:61:0D:DF:2F:A0:78:B5:C8:06:7F:4E:82:82:90:BF:B8:60:E8:4B:3
Alias name: mozillacert6.pem
  SHA1: 27:96:BA:E6:3F:18:01:E2:77:26:1B:A0:D7:77:70:02:8F:20:EE:E4
  SHA256:
 C3:84:6B:F2:4B:9E:93:CA:64:27:4C:0E:C6:7C:1E:CC:5E:02:4F:FC:AC:D2:D7:40:19:35:0E:81:FE:54:6A:E
Alias name: mozillacert60.pem
  SHA1: 3B:C4:9F:48:F8:F3:73:A0:9C:1E:BD:F8:5B:B1:C3:65:C7:D8:11:B3
  SHA256:
 BF:0F:EE:FB:9E:3A:58:1A:D5:F9:E9:DB:75:89:98:57:43:D2:61:08:5C:4D:31:4F:6F:5D:72:59:AA:42:16:1
Alias name: mozillacert61.pem
```

```
  SHA1: E0:B4:32:2E:B2:F6:A5:68:B6:54:53:84:48:18:4A:50:36:87:43:84
  SHA256:
 03:95:0F:B4:9A:53:1F:3E:19:91:94:23:98:DF:A9:E0:EA:32:D7:BA:1C:DD:9B:C8:5D:B5:7E:D9:40:0B:43:4
Alias name: mozillacert62.pem
  SHA1: A1:DB:63:93:91:6F:17:E4:18:55:09:40:04:15:C7:02:40:B0:AE:6B
  SHA256:
 A4:B6:B3:99:6F:C2:F3:06:B3:FD:86:81:BD:63:41:3D:8C:50:09:CC:4F:A3:29:C2:CC:F0:E2:FA:1B:14:03:0
Alias name: mozillacert63.pem
  SHA1: 89:DF:74:FE:5C:F4:0F:4A:80:F9:E3:37:7D:54:DA:91:E1:01:31:8E
  SHA256:
 3C:5F:81:FE:A5:FA:B8:2C:64:BF:A2:EA:EC:AF:CD:E8:E0:77:FC:86:20:A7:CA:E5:37:16:3D:F3:6E:DB:F3:7
Alias name: mozillacert64.pem
  SHA1: 62:7F:8D:78:27:65:63:99:D2:7D:7F:90:44:C9:FE:B3:F3:3E:FA:9A
  SHA256:
 AB:70:36:36:5C:71:54:AA:29:C2:C2:9F:5D:41:91:16:3B:16:2A:22:25:01:13:57:D5:6D:07:FF:A7:BC:1F:7
Alias name: mozillacert65.pem
  SHA1: 69:BD:8C:F4:9C:D3:00:FB:59:2E:17:93:CA:55:6A:F3:EC:AA:35:FB
  SHA256:
 BC:23:F9:8A:31:3C:B9:2D:E3:BB:FC:3A:5A:9F:44:61:AC:39:49:4C:4A:E1:5A:9E:9D:F1:31:E9:9B:73:01:9
Alias name: mozillacert66.pem
  SHA1: DD:E1:D2:A9:01:80:2E:1D:87:5E:84:B3:80:7E:4B:B1:FD:99:41:34
  SHA256:
 E6:09:07:84:65:A4:19:78:0C:B6:AC:4C:1C:0B:FB:46:53:D9:D9:CC:6E:B3:94:6E:B7:F3:D6:99:97:BA:D5:9
Alias name: mozillacert67.pem
  SHA1: D6:9B:56:11:48:F0:1C:77:C5:45:78:C1:09:26:DF:5B:85:69:76:AD
  SHA256:
 CB:B5:22:D7:B7:F1:27:AD:6A:01:13:86:5B:DF:1C:D4:10:2E:7D:07:59:AF:63:5A:7C:F4:72:0D:C9:63:C5:3
Alias name: mozillacert68.pem
  SHA1: AE:C5:FB:3F:C8:E1:BF:C4:E5:4F:03:07:5A:9A:E8:00:B7:F7:B6:FA
  SHA256:
 04:04:80:28:BF:1F:28:64:D4:8F:9A:D4:D8:32:94:36:6A:82:88:56:55:3F:3B:14:30:3F:90:14:7F:5D:40:E
Alias name: mozillacert69.pem
  SHA1: 2F:78:3D:25:52:18:A7:4A:65:39:71:B5:2C:A2:9C:45:15:6F:E9:19
  SHA256:
 25:30:CC:8E:98:32:15:02:BA:D9:6F:9B:1F:BA:1B:09:9E:2D:29:9E:0F:45:48:BB:91:4F:36:3B:C0:D4:53:1
Alias name: mozillacert7.pem
  SHA1: AD:7E:1C:28:B0:64:EF:8F:60:03:40:20:14:C3:D0:E3:37:0E:B5:8A
  SHA256:
 14:65:FA:20:53:97:B8:76:FA:A6:F0:A9:95:8E:55:90:E4:0F:CC:7F:AA:4F:B7:C2:C8:67:75:21:FB:5F:B6:5
Alias name: mozillacert70.pem
  SHA1: 78:6A:74:AC:76:AB:14:7F:9C:6A:30:50:BA:9E:A8:7E:FE:9A:CE:3C
  SHA256:
 06:3E:4A:FA:C4:91:DF:D3:32:F3:08:9B:85:42:E9:46:17:D8:93:D7:FE:94:4E:10:A7:93:7E:E2:9D:96:93:C
Alias name: mozillacert71.pem
```

```
  SHA1: 4A:BD:EE:EC:95:0D:35:9C:89:AE:C7:52:A1:2C:5B:29:F6:D6:AA:0C
  SHA256:
 13:63:35:43:93:34:A7:69:80:16:A0:D3:24:DE:72:28:4E:07:9D:7B:52:20:BB:8F:BD:74:78:16:EE:BE:BA:C
Alias name: mozillacert72.pem
  SHA1: 47:BE:AB:C9:22:EA:E8:0E:78:78:34:62:A7:9F:45:C2:54:FD:E6:8B
  SHA256:
 45:14:0B:32:47:EB:9C:C8:C5:B4:F0:D7:B5:30:91:F7:32:92:08:9E:6E:5A:63:E2:74:9D:D3:AC:A9:19:8E:D
Alias name: mozillacert73.pem
  SHA1: B5:1C:06:7C:EE:2B:0C:3D:F8:55:AB:2D:92:F4:FE:39:D4:E7:0F:0E
  SHA256:
 2C:E1:CB:0B:F9:D2:F9:E1:02:99:3F:BE:21:51:52:C3:B2:DD:0C:AB:DE:1C:68:E5:31:9B:83:91:54:DB:B7:F
Alias name: mozillacert74.pem
  SHA1: 92:5A:8F:8D:2C:6D:04:E0:66:5F:59:6A:FF:22:D8:63:E8:25:6F:3F
  SHA256:
 56:8D:69:05:A2:C8:87:08:A4:B3:02:51:90:ED:CF:ED:B1:97:4A:60:6A:13:C6:E5:29:0F:CB:2A:E6:3E:DA:B
Alias name: mozillacert75.pem
  SHA1: D2:32:09:AD:23:D3:14:23:21:74:E4:0D:7F:9D:62:13:97:86:63:3A
  SHA256:
 08:29:7A:40:47:DB:A2:36:80:C7:31:DB:6E:31:76:53:CA:78:48:E1:BE:BD:3A:0B:01:79:A7:07:F9:2C:F1:7
Alias name: mozillacert76.pem
  SHA1: F9:B5:B6:32:45:5F:9C:BE:EC:57:5F:80:DC:E9:6E:2C:C7:B2:78:B7
  SHA256:
 03:76:AB:1D:54:C5:F9:80:3C:E4:B2:E2:01:A0:EE:7E:EF:7B:57:B6:36:E8:A9:3C:9B:8D:48:60:C9:6F:5F:A
Alias name: mozillacert77.pem
  SHA1: 13:2D:0D:45:53:4B:69:97:CD:B2:D5:C3:39:E2:55:76:60:9B:5C:C6
  SHA256:
 EB:04:CF:5E:B1:F3:9A:FA:76:2F:2B:B1:20:F2:96:CB:A5:20:C1:B9:7D:B1:58:95:65:B8:1C:B9:A1:7B:72:4
Alias name: mozillacert78.pem
  SHA1: 29:36:21:02:8B:20:ED:02:F5:66:C5:32:D1:D6:ED:90:9F:45:00:2F
  SHA256:
 0A:81:EC:5A:92:97:77:F1:45:90:4A:F3:8D:5D:50:9F:66:B5:E2:C5:8F:CD:B5:31:05:8B:0E:17:F3:F0:B4:1
Alias name: mozillacert79.pem
  SHA1: D8:A6:33:2C:E0:03:6F:B1:85:F6:63:4F:7D:6A:06:65:26:32:28:27
  SHA256:
 70:A7:3F:7F:37:6B:60:07:42:48:90:45:34:B1:14:82:D5:BF:0E:69:8E:CC:49:8D:F5:25:77:EB:F2:E9:3B:9
Alias name: mozillacert8.pem
  SHA1: 3E:2B:F7:F2:03:1B:96:F3:8C:E6:C4:D8:A8:5D:3E:2D:58:47:6A:0F
  SHA256:
 C7:66:A9:BE:F2:D4:07:1C:86:3A:31:AA:49:20:E8:13:B2:D1:98:60:8C:B7:B7:CF:E2:11:43:B8:36:DF:09:E
Alias name: mozillacert80.pem
  SHA1: B8:23:6B:00:2F:1D:16:86:53:01:55:6C:11:A4:37:CA:EB:FF:C3:BB
  SHA256:
 BD:71:FD:F6:DA:97:E4:CF:62:D1:64:7A:DD:25:81:B0:7D:79:AD:F8:39:7E:B4:EC:BA:9C:5E:84:88:82:14:2
Alias name: mozillacert81.pem
```

```
  SHA1: 07:E0:32:E0:20:B7:2C:3F:19:2F:06:28:A2:59:3A:19:A7:0F:06:9E
  SHA256:
 5C:58:46:8D:55:F5:8E:49:7E:74:39:82:D2:B5:00:10:B6:D1:65:37:4A:CF:83:A7:D4:A3:2D:B7:68:C4:40:8
Alias name: mozillacert82.pem
  SHA1: 2E:14:DA:EC:28:F0:FA:1E:8E:38:9A:4E:AB:EB:26:C0:0A:D3:83:C3
  SHA256:
 FC:BF:E2:88:62:06:F7:2B:27:59:3C:8B:07:02:97:E1:2D:76:9E:D1:0E:D7:93:07:05:A8:09:8E:FF:C1:4D:1
Alias name: mozillacert83.pem
  SHA1: A0:73:E5:C5:BD:43:61:0D:86:4C:21:13:0A:85:58:57:CC:9C:EA:46
  SHA256:
 8C:4E:DF:D0:43:48:F3:22:96:9E:7E:29:A4:CD:4D:CA:00:46:55:06:1C:16:E1:B0:76:42:2E:F3:42:AD:63:0
Alias name: mozillacert84.pem
  SHA1: D3:C0:63:F2:19:ED:07:3E:34:AD:5D:75:0B:32:76:29:FF:D5:9A:F2
  SHA256:
 79:3C:BF:45:59:B9:FD:E3:8A:B2:2D:F1:68:69:F6:98:81:AE:14:C4:B0:13:9A:C7:88:A7:8A:1A:FC:CA:02:F
Alias name: mozillacert85.pem
  SHA1: CF:9E:87:6D:D3:EB:FC:42:26:97:A3:B5:A3:7A:A0:76:A9:06:23:48
  SHA256:
 BF:D8:8F:E1:10:1C:41:AE:3E:80:1B:F8:BE:56:35:0E:E9:BA:D1:A6:B9:BD:51:5E:DC:5C:6D:5B:87:11:AC:4
Alias name: mozillacert86.pem
  SHA1: 74:2C:31:92:E6:07:E4:24:EB:45:49:54:2B:E1:BB:C5:3E:61:74:E2
  SHA256:
 E7:68:56:34:EF:AC:F6:9A:CE:93:9A:6B:25:5B:7B:4F:AB:EF:42:93:5B:50:A2:65:AC:B5:CB:60:27:E4:4E:7
Alias name: mozillacert87.pem
  SHA1: 5F:3B:8C:F2:F8:10:B3:7D:78:B4:CE:EC:19:19:C3:73:34:B9:C7:74
  SHA256:
 51:3B:2C:EC:B8:10:D4:CD:E5:DD:85:39:1A:DF:C6:C2:DD:60:D8:7B:B7:36:D2:B5:21:48:4A:A4:7A:0E:BE:F
Alias name: mozillacert88.pem
  SHA1: FE:45:65:9B:79:03:5B:98:A1:61:B5:51:2E:AC:DA:58:09:48:22:4D
  SHA256:
 BC:10:4F:15:A4:8B:E7:09:DC:A5:42:A7:E1:D4:B9:DF:6F:05:45:27:E8:02:EA:A9:2D:59:54:44:25:8A:FE:7
Alias name: mozillacert89.pem
  SHA1: C8:EC:8C:87:92:69:CB:4B:AB:39:E9:8D:7E:57:67:F3:14:95:73:9D
  SHA256:
 E3:89:36:0D:0F:DB:AE:B3:D2:50:58:4B:47:30:31:4E:22:2F:39:C1:56:A0:20:14:4E:8D:96:05:61:79:15:0
Alias name: mozillacert9.pem
  SHA1: F4:8B:11:BF:DE:AB:BE:94:54:20:71:E6:41:DE:6B:BE:88:2B:40:B9
  SHA256:
 76:00:29:5E:EF:E8:5B:9E:1F:D6:24:DB:76:06:2A:AA:AE:59:81:8A:54:D2:77:4C:D4:C0:B2:C0:11:31:E1:B
Alias name: mozillacert90.pem
  SHA1: F3:73:B3:87:06:5A:28:84:8A:F2:F3:4A:CE:19:2B:DD:C7:8E:9C:AC
  SHA256:
 55:92:60:84:EC:96:3A:64:B9:6E:2A:BE:01:CE:0B:A8:6A:64:FB:FE:BC:C7:AA:B5:AF:C1:55:B3:7F:D7:60:6
Alias name: mozillacert91.pem
```

```
  SHA1: 3B:C0:38:0B:33:C3:F6:A6:0C:86:15:22:93:D9:DF:F5:4B:81:C0:04
  SHA256:
 C1:B4:82:99:AB:A5:20:8F:E9:63:0A:CE:55:CA:68:A0:3E:DA:5A:51:9C:88:02:A0:D3:A6:73:BE:8F:8E:55:7
Alias name: mozillacert92.pem
  SHA1: A3:F1:33:3F:E2:42:BF:CF:C5:D1:4E:8F:39:42:98:40:68:10:D1:A0
  SHA256:
 E1:78:90:EE:09:A3:FB:F4:F4:8B:9C:41:4A:17:D6:37:B7:A5:06:47:E9:BC:75:23:22:72:7F:CC:17:42:A9:1
Alias name: mozillacert93.pem
  SHA1: 31:F1:FD:68:22:63:20:EE:C6:3B:3F:9D:EA:4A:3E:53:7C:7C:39:17
  SHA256:
 C7:BA:65:67:DE:93:A7:98:AE:1F:AA:79:1E:71:2D:37:8F:AE:1F:93:C4:39:7F:EA:44:1B:B7:CB:E6:FD:59:9
Alias name: mozillacert94.pem
  SHA1: 49:0A:75:74:DE:87:0A:47:FE:58:EE:F6:C7:6B:EB:C6:0B:12:40:99
  SHA256:
 9A:11:40:25:19:7C:5B:B9:5D:94:E6:3D:55:CD:43:79:08:47:B6:46:B2:3C:DF:11:AD:A4:A0:0E:FF:15:FB:4
Alias name: mozillacert95.pem
  SHA1: DA:FA:F7:FA:66:84:EC:06:8F:14:50:BD:C7:C2:81:A5:BC:A9:64:57
  SHA256:
 ED:F7:EB:BC:A2:7A:2A:38:4D:38:7B:7D:40:10:C6:66:E2:ED:B4:84:3E:4C:29:B4:AE:1D:5B:93:32:E6:B2:4
Alias name: mozillacert96.pem
  SHA1: 55:A6:72:3E:CB:F2:EC:CD:C3:23:74:70:19:9D:2A:BE:11:E3:81:D1
  SHA256:
 FD:73:DA:D3:1C:64:4F:F1:B4:3B:EF:0C:CD:DA:96:71:0B:9C:D9:87:5E:CA:7E:31:70:7A:F3:E9:6D:52:2B:B
Alias name: mozillacert97.pem
  SHA1: 85:37:1C:A6:E5:50:14:3D:CE:28:03:47:1B:DE:3A:09:E8:F8:77:0F
  SHA256:
 83:CE:3C:12:29:68:8A:59:3D:48:5F:81:97:3C:0F:91:95:43:1E:DA:37:CC:5E:36:43:0E:79:C7:A8:88:63:8
Alias name: mozillacert98.pem
  SHA1: C9:A8:B9:E7:55:80:5E:58:E3:53:77:A7:25:EB:AF:C3:7B:27:CC:D7
  SHA256:
 3E:84:BA:43:42:90:85:16:E7:75:73:C0:99:2F:09:79:CA:08:4E:46:85:68:1F:F1:95:CC:BA:8A:22:9B:8A:7
Alias name: mozillacert99.pem
  SHA1: F1:7F:6F:B6:31:DC:99:E3:A3:C8:7F:FE:1C:F1:81:10:88:D9:60:33
  SHA256:
 97:8C:D9:66:F2:FA:A0:7B:A7:AA:95:00:D9:C0:2E:9D:77:F2:CD:AD:A6:AD:6B:A7:4A:F4:B9:1C:66:59:3C:5
Alias name: netlockaranyclassgoldfotanusitvany
  SHA1: 06:08:3F:59:3F:15:A1:04:A0:69:A4:6B:A9:03:D0:06:B7:97:09:91
  SHA256:
 6C:61:DA:C3:A2:DE:F0:31:50:6B:E0:36:D2:A6:FE:40:19:94:FB:D1:3D:F9:C8:D4:66:59:92:74:C4:46:EC:9
Alias name: networksolutionscertificateauthority
  SHA1: 74:F8:A3:C3:EF:E7:B3:90:06:4B:83:90:3C:21:64:60:20:E5:DF:CE
  SHA256:
 15:F0:BA:00:A3:AC:7A:F3:AC:88:4C:07:2B:10:11:A0:77:BD:77:C0:97:F4:01:64:B2:F8:59:8A:BD:83:86:0
Alias name: oistewisekeyglobalrootgaca
```

```
  SHA1: 59:22:A1:E1:5A:EA:16:35:21:F8:98:39:6A:46:46:B0:44:1B:0F:A9
  SHA256:
 41:C9:23:86:6A:B4:CA:D6:B7:AD:57:80:81:58:2E:02:07:97:A6:CB:DF:4F:FF:78:CE:83:96:B3:89:37:D7:F
Alias name: oistewisekeyglobalrootgbca
  SHA1: 0F:F9:40:76:18:D3:D7:6A:4B:98:F0:A8:35:9E:0C:FD:27:AC:CC:ED
  SHA256:
 6B:9C:08:E8:6E:B0:F7:67:CF:AD:65:CD:98:B6:21:49:E5:49:4A:67:F5:84:5E:7B:D1:ED:01:9F:27:B8:6B:D
Alias name: oistewisekeyglobalrootgcca
  SHA1: E0:11:84:5E:34:DE:BE:88:81:B9:9C:F6:16:26:D1:96:1F:C3:B9:31
  SHA256:
 85:60:F9:1C:36:24:DA:BA:95:70:B5:FE:A0:DB:E3:6F:F1:1A:83:23:BE:94:86:85:4F:B3:F3:4A:55:71:19:8
Alias name: quovadisrootca
  SHA1: DE:3F:40:BD:50:93:D3:9B:6C:60:F6:DA:BC:07:62:01:00:89:76:C9
  SHA256:
 A4:5E:DE:3B:BB:F0:9C:8A:E1:5C:72:EF:C0:72:68:D6:93:A2:1C:99:6F:D5:1E:67:CA:07:94:60:FD:6D:88:7
Alias name: quovadisrootca1g3
  SHA1: 1B:8E:EA:57:96:29:1A:C9:39:EA:B8:0A:81:1A:73:73:C0:93:79:67
  SHA256:
 8A:86:6F:D1:B2:76:B5:7E:57:8E:92:1C:65:82:8A:2B:ED:58:E9:F2:F2:88:05:41:34:B7:F1:F4:BF:C9:CC:7
Alias name: quovadisrootca2
  SHA1: CA:3A:FB:CF:12:40:36:4B:44:B2:16:20:88:80:48:39:19:93:7C:F7
  SHA256:
 85:A0:DD:7D:D7:20:AD:B7:FF:05:F8:3D:54:2B:20:9D:C7:FF:45:28:F7:D6:77:B1:83:89:FE:A5:E5:C4:9E:8
Alias name: quovadisrootca2g3
  SHA1: 09:3C:61:F3:8B:8B:DC:7D:55:DF:75:38:02:05:00:E1:25:F5:C8:36
  SHA256:
 8F:E4:FB:0A:F9:3A:4D:0D:67:DB:0B:EB:B2:3E:37:C7:1B:F3:25:DC:BC:DD:24:0E:A0:4D:AF:58:B4:7E:18:4
Alias name: quovadisrootca3
  SHA1: 1F:49:14:F7:D8:74:95:1D:DD:AE:02:C0:BE:FD:3A:2D:82:75:51:85
  SHA256:
 18:F1:FC:7F:20:5D:F8:AD:DD:EB:7F:E0:07:DD:57:E3:AF:37:5A:9C:4D:8D:73:54:6B:F4:F1:FE:D1:E1:8D:3
Alias name: quovadisrootca3g3
  SHA1: 48:12:BD:92:3C:A8:C4:39:06:E7:30:6D:27:96:E6:A4:CF:22:2E:7D
  SHA256:
 88:EF:81:DE:20:2E:B0:18:45:2E:43:F8:64:72:5C:EA:5F:BD:1F:C2:D9:D2:05:73:07:09:C5:D8:B8:69:0F:4
Alias name: secomevrootca1
  SHA1: FE:B8:C4:32:DC:F9:76:9A:CE:AE:3D:D8:90:8F:FD:28:86:65:64:7D
  SHA256:
 A2:2D:BA:68:1E:97:37:6E:2D:39:7D:72:8A:AE:3A:9B:62:96:B9:FD:BA:60:BC:2E:11:F6:47:F2:C6:75:FB:3
Alias name: secomscrootca1
  SHA1: 36:B1:2B:49:F9:81:9E:D7:4C:9E:BC:38:0F:C6:56:8F:5D:AC:B2:F7
  SHA256:
 E7:5E:72:ED:9F:56:0E:EC:6E:B4:80:00:73:A4:3F:C3:AD:19:19:5A:39:22:82:01:78:95:97:4A:99:02:6B:6
Alias name: secomscrootca2
```

```
  SHA1: 5F:3B:8C:F2:F8:10:B3:7D:78:B4:CE:EC:19:19:C3:73:34:B9:C7:74
  SHA256:
 51:3B:2C:EC:B8:10:D4:CD:E5:DD:85:39:1A:DF:C6:C2:DD:60:D8:7B:B7:36:D2:B5:21:48:4A:A4:7A:0E:BE:F
Alias name: secomvalicertclass1ca
  SHA1: E5:DF:74:3C:B6:01:C4:9B:98:43:DC:AB:8C:E8:6A:81:10:9F:E4:8E
  SHA256:
 F4:C1:49:55:1A:30:13:A3:5B:C7:BF:FE:17:A7:F3:44:9B:C1:AB:5B:5A:0A:E7:4B:06:C2:3B:90:00:4C:01:0
Alias name: secureglobalca
  SHA1: 3A:44:73:5A:E5:81:90:1F:24:86:61:46:1E:3B:9C:C4:5F:F5:3A:1B
  SHA256:
 42:00:F5:04:3A:C8:59:0E:BB:52:7D:20:9E:D1:50:30:29:FB:CB:D4:1C:A1:B5:06:EC:27:F1:5A:DE:7D:AC:6
Alias name: securesignrootca11
  SHA1: 3B:C4:9F:48:F8:F3:73:A0:9C:1E:BD:F8:5B:B1:C3:65:C7:D8:11:B3
  SHA256:
 BF:0F:EE:FB:9E:3A:58:1A:D5:F9:E9:DB:75:89:98:57:43:D2:61:08:5C:4D:31:4F:6F:5D:72:59:AA:42:16:1
Alias name: securetrustca
  SHA1: 87:82:C6:C3:04:35:3B:CF:D2:96:92:D2:59:3E:7D:44:D9:34:FF:11
  SHA256:
 F1:C1:B5:0A:E5:A2:0D:D8:03:0E:C9:F6:BC:24:82:3D:D3:67:B5:25:57:59:B4:E7:1B:61:FC:E9:F7:37:5D:7
Alias name: securitycommunicationrootca
  SHA1: 36:B1:2B:49:F9:81:9E:D7:4C:9E:BC:38:0F:C6:56:8F:5D:AC:B2:F7
  SHA256:
 E7:5E:72:ED:9F:56:0E:EC:6E:B4:80:00:73:A4:3F:C3:AD:19:19:5A:39:22:82:01:78:95:97:4A:99:02:6B:6
Alias name: securitycommunicationrootca2
  SHA1: 5F:3B:8C:F2:F8:10:B3:7D:78:B4:CE:EC:19:19:C3:73:34:B9:C7:74
  SHA256:
 51:3B:2C:EC:B8:10:D4:CD:E5:DD:85:39:1A:DF:C6:C2:DD:60:D8:7B:B7:36:D2:B5:21:48:4A:A4:7A:0E:BE:F
Alias name: soneraclass1ca
  SHA1: 07:47:22:01:99:CE:74:B9:7C:B0:3D:79:B2:64:A2:C8:55:E9:33:FF
  SHA256:
 CD:80:82:84:CF:74:6F:F2:FD:6E:B5:8A:A1:D5:9C:4A:D4:B3:CA:56:FD:C6:27:4A:89:26:A7:83:5F:32:31:3
Alias name: soneraclass2ca
  SHA1: 37:F7:6D:E6:07:7C:90:C5:B1:3E:93:1A:B7:41:10:B4:F2:E4:9A:27
  SHA256:
 79:08:B4:03:14:C1:38:10:0B:51:8D:07:35:80:7F:FB:FC:F8:51:8A:00:95:33:71:05:BA:38:6B:15:3D:D9:2
Alias name: soneraclass2rootca
  SHA1: 37:F7:6D:E6:07:7C:90:C5:B1:3E:93:1A:B7:41:10:B4:F2:E4:9A:27
  SHA256:
 79:08:B4:03:14:C1:38:10:0B:51:8D:07:35:80:7F:FB:FC:F8:51:8A:00:95:33:71:05:BA:38:6B:15:3D:D9:2
Alias name: sslcomevrootcertificationauthorityecc
  SHA1: 4C:DD:51:A3:D1:F5:20:32:14:B0:C6:C5:32:23:03:91:C7:46:42:6D
  SHA256:
 22:A2:C1:F7:BD:ED:70:4C:C1:E7:01:B5:F4:08:C3:10:88:0F:E9:56:B5:DE:2A:4A:44:F9:9C:87:3A:25:A7:C
Alias name: sslcomevrootcertificationauthorityrsar2
```

```
  SHA1: 74:3A:F0:52:9B:D0:32:A0:F4:4A:83:CD:D4:BA:A9:7B:7C:2E:C4:9A
  SHA256:
 2E:7B:F1:6C:C2:24:85:A7:BB:E2:AA:86:96:75:07:61:B0:AE:39:BE:3B:2F:E9:D0:CC:6D:4E:F7:34:91:42:5
Alias name: sslcomrootcertificationauthorityecc
  SHA1: C3:19:7C:39:24:E6:54:AF:1B:C4:AB:20:95:7A:E2:C3:0E:13:02:6A
  SHA256:
 34:17:BB:06:CC:60:07:DA:1B:96:1C:92:0B:8A:B4:CE:3F:AD:82:0E:4A:A3:0B:9A:CB:C4:A7:4E:BD:CE:BC:6
Alias name: sslcomrootcertificationauthorityrsa
  SHA1: B7:AB:33:08:D1:EA:44:77:BA:14:80:12:5A:6F:BD:A9:36:49:0C:BB
  SHA256:
 85:66:6A:56:2E:E0:BE:5C:E9:25:C1:D8:89:0A:6F:76:A8:7E:C1:6D:4D:7D:5F:29:EA:74:19:CF:20:12:3B:6
Alias name: staatdernederlandenevrootca
  SHA1: 76:E2:7E:C1:4F:DB:82:C1:C0:A6:75:B5:05:BE:3D:29:B4:ED:DB:BB
  SHA256:
 4D:24:91:41:4C:FE:95:67:46:EC:4C:EF:A6:CF:6F:72:E2:8A:13:29:43:2F:9D:8A:90:7A:C4:CB:5D:AD:C1:5
Alias name: staatdernederlandenrootcag3
  SHA1: D8:EB:6B:41:51:92:59:E0:F3:E7:85:00:C0:3D:B6:88:97:C9:EE:FC
  SHA256:
 3C:4F:B0:B9:5A:B8:B3:00:32:F4:32:B8:6F:53:5F:E1:72:C1:85:D0:FD:39:86:58:37:CF:36:18:7F:A6:F4:2
Alias name: starfieldclass2ca
  SHA1: AD:7E:1C:28:B0:64:EF:8F:60:03:40:20:14:C3:D0:E3:37:0E:B5:8A
  SHA256:
 14:65:FA:20:53:97:B8:76:FA:A6:F0:A9:95:8E:55:90:E4:0F:CC:7F:AA:4F:B7:C2:C8:67:75:21:FB:5F:B6:5
Alias name: starfieldrootcertificateauthorityg2
  SHA1: B5:1C:06:7C:EE:2B:0C:3D:F8:55:AB:2D:92:F4:FE:39:D4:E7:0F:0E
  SHA256:
 2C:E1:CB:0B:F9:D2:F9:E1:02:99:3F:BE:21:51:52:C3:B2:DD:0C:AB:DE:1C:68:E5:31:9B:83:91:54:DB:B7:F
Alias name: starfieldrootg2ca
  SHA1: B5:1C:06:7C:EE:2B:0C:3D:F8:55:AB:2D:92:F4:FE:39:D4:E7:0F:0E
  SHA256:
 2C:E1:CB:0B:F9:D2:F9:E1:02:99:3F:BE:21:51:52:C3:B2:DD:0C:AB:DE:1C:68:E5:31:9B:83:91:54:DB:B7:F
Alias name: starfieldservicesrootcertificateauthorityg2
  SHA1: 92:5A:8F:8D:2C:6D:04:E0:66:5F:59:6A:FF:22:D8:63:E8:25:6F:3F
  SHA256:
 56:8D:69:05:A2:C8:87:08:A4:B3:02:51:90:ED:CF:ED:B1:97:4A:60:6A:13:C6:E5:29:0F:CB:2A:E6:3E:DA:B
Alias name: starfieldservicesrootg2ca
  SHA1: 92:5A:8F:8D:2C:6D:04:E0:66:5F:59:6A:FF:22:D8:63:E8:25:6F:3F
  SHA256:
 56:8D:69:05:A2:C8:87:08:A4:B3:02:51:90:ED:CF:ED:B1:97:4A:60:6A:13:C6:E5:29:0F:CB:2A:E6:3E:DA:B
Alias name: swisssigngoldcag2
  SHA1: D8:C5:38:8A:B7:30:1B:1B:6E:D4:7A:E6:45:25:3A:6F:9F:1A:27:61
  SHA256:
 62:DD:0B:E9:B9:F5:0A:16:3E:A0:F8:E7:5C:05:3B:1E:CA:57:EA:55:C8:68:8F:64:7C:68:81:F2:C8:35:7B:9
Alias name: swisssigngoldg2ca
```

```
  SHA1: D8:C5:38:8A:B7:30:1B:1B:6E:D4:7A:E6:45:25:3A:6F:9F:1A:27:61
  SHA256:
 62:DD:0B:E9:B9:F5:0A:16:3E:A0:F8:E7:5C:05:3B:1E:CA:57:EA:55:C8:68:8F:64:7C:68:81:F2:C8:35:7B:9
Alias name: swisssignplatinumg2ca
  SHA1: 56:E0:FA:C0:3B:8F:18:23:55:18:E5:D3:11:CA:E8:C2:43:31:AB:66
  SHA256:
 3B:22:2E:56:67:11:E9:92:30:0D:C0:B1:5A:B9:47:3D:AF:DE:F8:C8:4D:0C:EF:7D:33:17:B4:C1:82:1D:14:3
Alias name: swisssignsilvercag2
  SHA1: 9B:AA:E5:9F:56:EE:21:CB:43:5A:BE:25:93:DF:A7:F0:40:D1:1D:CB
  SHA256:
 BE:6C:4D:A2:BB:B9:BA:59:B6:F3:93:97:68:37:42:46:C3:C0:05:99:3F:A9:8F:02:0D:1D:ED:BE:D4:8A:81:D
Alias name: swisssignsilverg2ca
  SHA1: 9B:AA:E5:9F:56:EE:21:CB:43:5A:BE:25:93:DF:A7:F0:40:D1:1D:CB
  SHA256:
 BE:6C:4D:A2:BB:B9:BA:59:B6:F3:93:97:68:37:42:46:C3:C0:05:99:3F:A9:8F:02:0D:1D:ED:BE:D4:8A:81:D
Alias name: szafirrootca2
  SHA1: E2:52:FA:95:3F:ED:DB:24:60:BD:6E:28:F3:9C:CC:CF:5E:B3:3F:DE
  SHA256:
 A1:33:9D:33:28:1A:0B:56:E5:57:D3:D3:2B:1C:E7:F9:36:7E:B0:94:BD:5F:A7:2A:7E:50:04:C8:DE:D7:CA:F
Alias name: teliasonerarootcav1
  SHA1: 43:13:BB:96:F1:D5:86:9B:C1:4E:6A:92:F6:CF:F6:34:69:87:82:37
  SHA256:
 DD:69:36:FE:21:F8:F0:77:C1:23:A1:A5:21:C1:22:24:F7:22:55:B7:3E:03:A7:26:06:93:E8:A2:4B:0F:A3:8
Alias name: thawtepersonalfreemailca
  SHA1: E6:18:83:AE:84:CA:C1:C1:CD:52:AD:E8:E9:25:2B:45:A6:4F:B7:E2
  SHA256:
 5B:38:BD:12:9E:83:D5:A0:CA:D2:39:21:08:94:90:D5:0D:4A:AE:37:04:28:F8:DD:FF:FF:FA:4C:15:64:E1:8
Alias name: thawtepremiumserverca
  SHA1: E0:AB:05:94:20:72:54:93:05:60:62:02:36:70:F7:CD:2E:FC:66:66
  SHA256:
 3F:9F:27:D5:83:20:4B:9E:09:C8:A3:D2:06:6C:4B:57:D3:A2:47:9C:36:93:65:08:80:50:56:98:10:5D:BC:E
Alias name: thawteprimaryrootca
  SHA1: 91:C6:D6:EE:3E:8A:C8:63:84:E5:48:C2:99:29:5C:75:6C:81:7B:81
  SHA256:
 8D:72:2F:81:A9:C1:13:C0:79:1D:F1:36:A2:96:6D:B2:6C:95:0A:97:1D:B4:6B:41:99:F4:EA:54:B7:8B:FB:9
Alias name: thawteprimaryrootcag2
  SHA1: AA:DB:BC:22:23:8F:C4:01:A1:27:BB:38:DD:F4:1D:DB:08:9E:F0:12
  SHA256:
 A4:31:0D:50:AF:18:A6:44:71:90:37:2A:86:AF:AF:8B:95:1F:FB:43:1D:83:7F:1E:56:88:B4:59:71:ED:15:5
Alias name: thawteprimaryrootcag3
  SHA1: F1:8B:53:8D:1B:E9:03:B6:A6:F0:56:43:5B:17:15:89:CA:F3:6B:F2
  SHA256:
 4B:03:F4:58:07:AD:70:F2:1B:FC:2C:AE:71:C9:FD:E4:60:4C:06:4C:F5:FF:B6:86:BA:E5:DB:AA:D7:FD:D3:4
Alias name: thawteserverca
```

```
  SHA1: 9F:AD:91:A6:CE:6A:C6:C5:00:47:C4:4E:C9:D4:A5:0D:92:D8:49:79
  SHA256:
 87:C6:78:BF:B8:B2:5F:38:F7:E9:7B:33:69:56:BB:CF:14:4B:BA:CA:A5:36:47:E6:1A:23:25:BC:10:55:31:6
Alias name: trustcenterclass2caii
  SHA1: AE:50:83:ED:7C:F4:5C:BC:8F:61:C6:21:FE:68:5D:79:42:21:15:6E
  SHA256:
 E6:B8:F8:76:64:85:F8:07:AE:7F:8D:AC:16:70:46:1F:07:C0:A1:3E:EF:3A:1F:F7:17:53:8D:7A:BA:D3:91:B
Alias name: trustcenterclass4caii
  SHA1: A6:9A:91:FD:05:7F:13:6A:42:63:0B:B1:76:0D:2D:51:12:0C:16:50
  SHA256:
 32:66:96:7E:59:CD:68:00:8D:9D:D3:20:81:11:85:C7:04:20:5E:8D:95:FD:D8:4F:1C:7B:31:1E:67:04:FC:3
Alias name: trustcenteruniversalcai
  SHA1: 6B:2F:34:AD:89:58:BE:62:FD:B0:6B:5C:CE:BB:9D:D9:4F:4E:39:F3
  SHA256:
 EB:F3:C0:2A:87:89:B1:FB:7D:51:19:95:D6:63:B7:29:06:D9:13:CE:0D:5E:10:56:8A:8A:77:E2:58:61:67:E
Alias name: trustcoreca1
  SHA1: 58:D1:DF:95:95:67:6B:63:C0:F0:5B:1C:17:4D:8B:84:0B:C8:78:BD
  SHA256:
 5A:88:5D:B1:9C:01:D9:12:C5:75:93:88:93:8C:AF:BB:DF:03:1A:B2:D4:8E:91:EE:15:58:9B:42:97:1D:03:9
Alias name: trustcorrootcertca1
  SHA1: FF:BD:CD:E7:82:C8:43:5E:3C:6F:26:86:5C:CA:A8:3A:45:5B:C3:0A
  SHA256:
 D4:0E:9C:86:CD:8F:E4:68:C1:77:69:59:F4:9E:A7:74:FA:54:86:84:B6:C4:06:F3:90:92:61:F4:DC:E2:57:5
Alias name: trustcorrootcertca2
  SHA1: B8:BE:6D:CB:56:F1:55:B9:63:D4:12:CA:4E:06:34:C7:94:B2:1C:C0
  SHA256:
 07:53:E9:40:37:8C:1B:D5:E3:83:6E:39:5D:AE:A5:CB:83:9E:50:46:F1:BD:0E:AE:19:51:CF:10:FE:C7:C9:6
Alias name: trustisfpsrootca
  SHA1: 3B:C0:38:0B:33:C3:F6:A6:0C:86:15:22:93:D9:DF:F5:4B:81:C0:04
  SHA256:
 C1:B4:82:99:AB:A5:20:8F:E9:63:0A:CE:55:CA:68:A0:3E:DA:5A:51:9C:88:02:A0:D3:A6:73:BE:8F:8E:55:7
Alias name: ttelesecglobalrootclass2
  SHA1: 59:0D:2D:7D:88:4F:40:2E:61:7E:A5:62:32:17:65:CF:17:D8:94:E9
  SHA256:
 91:E2:F5:78:8D:58:10:EB:A7:BA:58:73:7D:E1:54:8A:8E:CA:CD:01:45:98:BC:0B:14:3E:04:1B:17:05:25:5
Alias name: ttelesecglobalrootclass2ca
  SHA1: 59:0D:2D:7D:88:4F:40:2E:61:7E:A5:62:32:17:65:CF:17:D8:94:E9
  SHA256:
 91:E2:F5:78:8D:58:10:EB:A7:BA:58:73:7D:E1:54:8A:8E:CA:CD:01:45:98:BC:0B:14:3E:04:1B:17:05:25:5
Alias name: ttelesecglobalrootclass3
  SHA1: 55:A6:72:3E:CB:F2:EC:CD:C3:23:74:70:19:9D:2A:BE:11:E3:81:D1
  SHA256:
 FD:73:DA:D3:1C:64:4F:F1:B4:3B:EF:0C:CD:DA:96:71:0B:9C:D9:87:5E:CA:7E:31:70:7A:F3:E9:6D:52:2B:B
Alias name: ttelesecglobalrootclass3ca
```

```
  SHA1: 55:A6:72:3E:CB:F2:EC:CD:C3:23:74:70:19:9D:2A:BE:11:E3:81:D1
  SHA256:
 FD:73:DA:D3:1C:64:4F:F1:B4:3B:EF:0C:CD:DA:96:71:0B:9C:D9:87:5E:CA:7E:31:70:7A:F3:E9:6D:52:2B:B
Alias name: tubitakkamusmsslkoksertifikasisurum1
  SHA1: 31:43:64:9B:EC:CE:27:EC:ED:3A:3F:0B:8F:0D:E4:E8:91:DD:EE:CA
  SHA256:
 46:ED:C3:68:90:46:D5:3A:45:3F:B3:10:4A:B8:0D:CA:EC:65:8B:26:60:EA:16:29:DD:7E:86:79:90:64:87:1
Alias name: twcaglobalrootca
  SHA1: 9C:BB:48:53:F6:A4:F6:D3:52:A4:E8:32:52:55:60:13:F5:AD:AF:65
  SHA256:
 59:76:90:07:F7:68:5D:0F:CD:50:87:2F:9F:95:D5:75:5A:5B:2B:45:7D:81:F3:69:2B:61:0A:98:67:2F:0E:1
Alias name: twcarootcertificationauthority
  SHA1: CF:9E:87:6D:D3:EB:FC:42:26:97:A3:B5:A3:7A:A0:76:A9:06:23:48
  SHA256:
 BF:D8:8F:E1:10:1C:41:AE:3E:80:1B:F8:BE:56:35:0E:E9:BA:D1:A6:B9:BD:51:5E:DC:5C:6D:5B:87:11:AC:4
Alias name: ucaextendedvalidationroot
  SHA1: A3:A1:B0:6F:24:61:23:4A:E3:36:A5:C2:37:FC:A6:FF:DD:F0:D7:3A
  SHA256:
 D4:3A:F9:B3:54:73:75:5C:96:84:FC:06:D7:D8:CB:70:EE:5C:28:E7:73:FB:29:4E:B4:1E:E7:17:22:92:4D:2
Alias name: ucaglobalg2root
  SHA1: 28:F9:78:16:19:7A:FF:18:25:18:AA:44:FE:C1:A0:CE:5C:B6:4C:8A
  SHA256:
 9B:EA:11:C9:76:FE:01:47:64:C1:BE:56:A6:F9:14:B5:A5:60:31:7A:BD:99:88:39:33:82:E5:16:1A:A0:49:3
Alias name: usertrustecc
  SHA1: D1:CB:CA:5D:B2:D5:2A:7F:69:3B:67:4D:E5:F0:5A:1D:0C:95:7D:F0
  SHA256:
 4F:F4:60:D5:4B:9C:86:DA:BF:BC:FC:57:12:E0:40:0D:2B:ED:3F:BC:4D:4F:BD:AA:86:E0:6A:DC:D2:A9:AD:7
Alias name: usertrustecccertificationauthority
  SHA1: D1:CB:CA:5D:B2:D5:2A:7F:69:3B:67:4D:E5:F0:5A:1D:0C:95:7D:F0
  SHA256:
 4F:F4:60:D5:4B:9C:86:DA:BF:BC:FC:57:12:E0:40:0D:2B:ED:3F:BC:4D:4F:BD:AA:86:E0:6A:DC:D2:A9:AD:7
Alias name: usertrustrsa
  SHA1: 2B:8F:1B:57:33:0D:BB:A2:D0:7A:6C:51:F7:0E:E9:0D:DA:B9:AD:8E
  SHA256:
 E7:93:C9:B0:2F:D8:AA:13:E2:1C:31:22:8A:CC:B0:81:19:64:3B:74:9C:89:89:64:B1:74:6D:46:C3:D4:CB:D
Alias name: usertrustrsacertificationauthority
  SHA1: 2B:8F:1B:57:33:0D:BB:A2:D0:7A:6C:51:F7:0E:E9:0D:DA:B9:AD:8E
  SHA256:
 E7:93:C9:B0:2F:D8:AA:13:E2:1C:31:22:8A:CC:B0:81:19:64:3B:74:9C:89:89:64:B1:74:6D:46:C3:D4:CB:D
Alias name: utndatacorpsgcca
  SHA1: 58:11:9F:0E:12:82:87:EA:50:FD:D9:87:45:6F:4F:78:DC:FA:D6:D4
  SHA256:
 85:FB:2F:91:DD:12:27:5A:01:45:B6:36:53:4F:84:02:4A:D6:8B:69:B8:EE:88:68:4F:F7:11:37:58:05:B3:4
Alias name: utnuserfirstclientauthemailca
```

```
  SHA1: B1:72:B1:A5:6D:95:F9:1F:E5:02:87:E1:4D:37:EA:6A:44:63:76:8A
  SHA256:
 43:F2:57:41:2D:44:0D:62:74:76:97:4F:87:7D:A8:F1:FC:24:44:56:5A:36:7A:E6:0E:DD:C2:7A:41:25:31:A
Alias name: utnuserfirsthardwareca
  SHA1: 04:83:ED:33:99:AC:36:08:05:87:22:ED:BC:5E:46:00:E3:BE:F9:D7
  SHA256:
 6E:A5:47:41:D0:04:66:7E:ED:1B:48:16:63:4A:A3:A7:9E:6E:4B:96:95:0F:82:79:DA:FC:8D:9B:D8:81:21:3
Alias name: utnuserfirstobjectca
  SHA1: E1:2D:FB:4B:41:D7:D9:C3:2B:30:51:4B:AC:1D:81:D8:38:5E:2D:46
  SHA256:
 6F:FF:78:E4:00:A7:0C:11:01:1C:D8:59:77:C4:59:FB:5A:F9:6A:3D:F0:54:08:20:D0:F4:B8:60:78:75:E5:8
Alias name: valicertclass2ca
  SHA1: 31:7A:2A:D0:7F:2B:33:5E:F5:A1:C3:4E:4B:57:E8:B7:D8:F1:FC:A6
  SHA256:
 58:D0:17:27:9C:D4:DC:63:AB:DD:B1:96:A6:C9:90:6C:30:C4:E0:87:83:EA:E8:C1:60:99:54:D6:93:55:59:6
Alias name: verisignc1g1.pem
  SHA1: 90:AE:A2:69:85:FF:14:80:4C:43:49:52:EC:E9:60:84:77:AF:55:6F
  SHA256:
 D1:7C:D8:EC:D5:86:B7:12:23:8A:48:2C:E4:6F:A5:29:39:70:74:2F:27:6D:8A:B6:A9:E4:6E:E0:28:8F:33:5
Alias name: verisignc1g2.pem
  SHA1: 27:3E:E1:24:57:FD:C4:F9:0C:55:E8:2B:56:16:7F:62:F5:32:E5:47
  SHA256:
 34:1D:E9:8B:13:92:AB:F7:F4:AB:90:A9:60:CF:25:D4:BD:6E:C6:5B:9A:51:CE:6E:D0:67:D0:0E:C7:CE:9B:7
Alias name: verisignc1g3.pem
  SHA1: 20:42:85:DC:F7:EB:76:41:95:57:8E:13:6B:D4:B7:D1:E9:8E:46:A5
  SHA256:
 CB:B5:AF:18:5E:94:2A:24:02:F9:EA:CB:C0:ED:5B:B8:76:EE:A3:C1:22:36:23:D0:04:47:E4:F3:BA:55:4B:6
Alias name: verisignc1g6.pem
  SHA1: 51:7F:61:1E:29:91:6B:53:82:FB:72:E7:44:D9:8D:C3:CC:53:6D:64
  SHA256:
 9D:19:0B:2E:31:45:66:68:5B:E8:A8:89:E2:7A:A8:C7:D7:AE:1D:8A:AD:DB:A3:C1:EC:F9:D2:48:63:CD:34:B
Alias name: verisignc2g1.pem
  SHA1: 67:82:AA:E0:ED:EE:E2:1A:58:39:D3:C0:CD:14:68:0A:4F:60:14:2A
  SHA256:
 BD:46:9F:F4:5F:AA:E7:C5:4C:CB:D6:9D:3F:3B:00:22:55:D9:B0:6B:10:B1:D0:FA:38:8B:F9:6B:91:8B:2C:E
Alias name: verisignc2g2.pem
  SHA1: B3:EA:C4:47:76:C9:C8:1C:EA:F2:9D:95:B6:CC:A0:08:1B:67:EC:9D
  SHA256:
 3A:43:E2:20:FE:7F:3E:A9:65:3D:1E:21:74:2E:AC:2B:75:C2:0F:D8:98:03:05:BC:50:2C:AF:8C:2D:9B:41:A
Alias name: verisignc2g3.pem
  SHA1: 61:EF:43:D7:7F:CA:D4:61:51:BC:98:E0:C3:59:12:AF:9F:EB:63:11
  SHA256:
 92:A9:D9:83:3F:E1:94:4D:B3:66:E8:BF:AE:7A:95:B6:48:0C:2D:6C:6C:2A:1B:E6:5D:42:36:B6:08:FC:A1:B
Alias name: verisignc2g6.pem
```

```
  SHA1: 40:B3:31:A0:E9:BF:E8:55:BC:39:93:CA:70:4F:4E:C2:51:D4:1D:8F
  SHA256:
 CB:62:7D:18:B5:8A:D5:6D:DE:33:1A:30:45:6B:C6:5C:60:1A:4E:9B:18:DE:DC:EA:08:E7:DA:AA:07:81:5F:F
Alias name: verisignc3g1.pem
  SHA1: A1:DB:63:93:91:6F:17:E4:18:55:09:40:04:15:C7:02:40:B0:AE:6B
  SHA256:
 A4:B6:B3:99:6F:C2:F3:06:B3:FD:86:81:BD:63:41:3D:8C:50:09:CC:4F:A3:29:C2:CC:F0:E2:FA:1B:14:03:0
Alias name: verisignc3g2.pem
  SHA1: 85:37:1C:A6:E5:50:14:3D:CE:28:03:47:1B:DE:3A:09:E8:F8:77:0F
  SHA256:
 83:CE:3C:12:29:68:8A:59:3D:48:5F:81:97:3C:0F:91:95:43:1E:DA:37:CC:5E:36:43:0E:79:C7:A8:88:63:8
Alias name: verisignc3g3.pem
  SHA1: 13:2D:0D:45:53:4B:69:97:CD:B2:D5:C3:39:E2:55:76:60:9B:5C:C6
  SHA256:
 EB:04:CF:5E:B1:F3:9A:FA:76:2F:2B:B1:20:F2:96:CB:A5:20:C1:B9:7D:B1:58:95:65:B8:1C:B9:A1:7B:72:4
Alias name: verisignc3g4.pem
  SHA1: 22:D5:D8:DF:8F:02:31:D1:8D:F7:9D:B7:CF:8A:2D:64:C9:3F:6C:3A
  SHA256:
 69:DD:D7:EA:90:BB:57:C9:3E:13:5D:C8:5E:A6:FC:D5:48:0B:60:32:39:BD:C4:54:FC:75:8B:2A:26:CF:7F:7
Alias name: verisignc3g5.pem
  SHA1: 4E:B6:D5:78:49:9B:1C:CF:5F:58:1E:AD:56:BE:3D:9B:67:44:A5:E5
  SHA256:
 9A:CF:AB:7E:43:C8:D8:80:D0:6B:26:2A:94:DE:EE:E4:B4:65:99:89:C3:D0:CA:F1:9B:AF:64:05:E4:1A:B7:D
Alias name: verisignc4g2.pem
  SHA1: 0B:77:BE:BB:CB:7A:A2:47:05:DE:CC:0F:BD:6A:02:FC:7A:BD:9B:52
  SHA256:
 44:64:0A:0A:0E:4D:00:0F:BD:57:4D:2B:8A:07:BD:B4:D1:DF:ED:3B:45:BA:AB:A7:6F:78:57:78:C7:01:19:6
Alias name: verisignc4g3.pem
  SHA1: C8:EC:8C:87:92:69:CB:4B:AB:39:E9:8D:7E:57:67:F3:14:95:73:9D
  SHA256:
 E3:89:36:0D:0F:DB:AE:B3:D2:50:58:4B:47:30:31:4E:22:2F:39:C1:56:A0:20:14:4E:8D:96:05:61:79:15:0
Alias name: verisignclass1ca
  SHA1: CE:6A:64:A3:09:E4:2F:BB:D9:85:1C:45:3E:64:09:EA:E8:7D:60:F1
  SHA256:
 51:84:7C:8C:BD:2E:9A:72:C9:1E:29:2D:2A:E2:47:D7:DE:1E:3F:D2:70:54:7A:20:EF:7D:61:0F:38:B8:84:2
Alias name: verisignclass1g2ca
  SHA1: 27:3E:E1:24:57:FD:C4:F9:0C:55:E8:2B:56:16:7F:62:F5:32:E5:47
  SHA256:
 34:1D:E9:8B:13:92:AB:F7:F4:AB:90:A9:60:CF:25:D4:BD:6E:C6:5B:9A:51:CE:6E:D0:67:D0:0E:C7:CE:9B:7
Alias name: verisignclass1g3ca
  SHA1: 20:42:85:DC:F7:EB:76:41:95:57:8E:13:6B:D4:B7:D1:E9:8E:46:A5
  SHA256:
 CB:B5:AF:18:5E:94:2A:24:02:F9:EA:CB:C0:ED:5B:B8:76:EE:A3:C1:22:36:23:D0:04:47:E4:F3:BA:55:4B:6
Alias name: verisignclass2g2ca
```

```
  SHA1: B3:EA:C4:47:76:C9:C8:1C:EA:F2:9D:95:B6:CC:A0:08:1B:67:EC:9D
  SHA256:
 3A:43:E2:20:FE:7F:3E:A9:65:3D:1E:21:74:2E:AC:2B:75:C2:0F:D8:98:03:05:BC:50:2C:AF:8C:2D:9B:41:A
Alias name: verisignclass2g3ca
  SHA1: 61:EF:43:D7:7F:CA:D4:61:51:BC:98:E0:C3:59:12:AF:9F:EB:63:11
  SHA256:
 92:A9:D9:83:3F:E1:94:4D:B3:66:E8:BF:AE:7A:95:B6:48:0C:2D:6C:6C:2A:1B:E6:5D:42:36:B6:08:FC:A1:B
Alias name: verisignclass3ca
  SHA1: A1:DB:63:93:91:6F:17:E4:18:55:09:40:04:15:C7:02:40:B0:AE:6B
  SHA256:
 A4:B6:B3:99:6F:C2:F3:06:B3:FD:86:81:BD:63:41:3D:8C:50:09:CC:4F:A3:29:C2:CC:F0:E2:FA:1B:14:03:0
Alias name: verisignclass3g2ca
  SHA1: 85:37:1C:A6:E5:50:14:3D:CE:28:03:47:1B:DE:3A:09:E8:F8:77:0F
  SHA256:
 83:CE:3C:12:29:68:8A:59:3D:48:5F:81:97:3C:0F:91:95:43:1E:DA:37:CC:5E:36:43:0E:79:C7:A8:88:63:8
Alias name: verisignclass3g3ca
  SHA1: 13:2D:0D:45:53:4B:69:97:CD:B2:D5:C3:39:E2:55:76:60:9B:5C:C6
  SHA256:
 EB:04:CF:5E:B1:F3:9A:FA:76:2F:2B:B1:20:F2:96:CB:A5:20:C1:B9:7D:B1:58:95:65:B8:1C:B9:A1:7B:72:4
Alias name: verisignclass3g4ca
  SHA1: 22:D5:D8:DF:8F:02:31:D1:8D:F7:9D:B7:CF:8A:2D:64:C9:3F:6C:3A
  SHA256:
 69:DD:D7:EA:90:BB:57:C9:3E:13:5D:C8:5E:A6:FC:D5:48:0B:60:32:39:BD:C4:54:FC:75:8B:2A:26:CF:7F:7
Alias name: verisignclass3g5ca
  SHA1: 4E:B6:D5:78:49:9B:1C:CF:5F:58:1E:AD:56:BE:3D:9B:67:44:A5:E5
  SHA256:
 9A:CF:AB:7E:43:C8:D8:80:D0:6B:26:2A:94:DE:EE:E4:B4:65:99:89:C3:D0:CA:F1:9B:AF:64:05:E4:1A:B7:D
Alias name: verisignclass3publicprimarycertificationauthorityg4
  SHA1: 22:D5:D8:DF:8F:02:31:D1:8D:F7:9D:B7:CF:8A:2D:64:C9:3F:6C:3A
  SHA256:
 69:DD:D7:EA:90:BB:57:C9:3E:13:5D:C8:5E:A6:FC:D5:48:0B:60:32:39:BD:C4:54:FC:75:8B:2A:26:CF:7F:7
Alias name: verisignclass3publicprimarycertificationauthorityg5
  SHA1: 4E:B6:D5:78:49:9B:1C:CF:5F:58:1E:AD:56:BE:3D:9B:67:44:A5:E5
  SHA256:
 9A:CF:AB:7E:43:C8:D8:80:D0:6B:26:2A:94:DE:EE:E4:B4:65:99:89:C3:D0:CA:F1:9B:AF:64:05:E4:1A:B7:D
Alias name: verisignroot.pem
  SHA1: 36:79:CA:35:66:87:72:30:4D:30:A5:FB:87:3B:0F:A7:7B:B7:0D:54
  SHA256:
 23:99:56:11:27:A5:71:25:DE:8C:EF:EA:61:0D:DF:2F:A0:78:B5:C8:06:7F:4E:82:82:90:BF:B8:60:E8:4B:3
Alias name: verisigntsaca
  SHA1: 20:CE:B1:F0:F5:1C:0E:19:A9:F3:8D:B1:AA:8E:03:8C:AA:7A:C7:01
  SHA256:
 CB:6B:05:D9:E8:E5:7C:D8:82:B1:0B:4D:B7:0D:E4:BB:1D:E4:2B:A4:8A:7B:D0:31:8B:63:5B:F6:E7:78:1A:9
Alias name: verisignuniversalrootca
```

```
  SHA1: 36:79:CA:35:66:87:72:30:4D:30:A5:FB:87:3B:0F:A7:7B:B7:0D:54
  SHA256:
 23:99:56:11:27:A5:71:25:DE:8C:EF:EA:61:0D:DF:2F:A0:78:B5:C8:06:7F:4E:82:82:90:BF:B8:60:E8:4B:3
Alias name: verisignuniversalrootcertificationauthority
  SHA1: 36:79:CA:35:66:87:72:30:4D:30:A5:FB:87:3B:0F:A7:7B:B7:0D:54
  SHA256:
 23:99:56:11:27:A5:71:25:DE:8C:EF:EA:61:0D:DF:2F:A0:78:B5:C8:06:7F:4E:82:82:90:BF:B8:60:E8:4B:3
Alias name: xrampglobalca
  SHA1: B8:01:86:D1:EB:9C:86:A5:41:04:CF:30:54:F3:4C:52:B7:E5:58:C6
  SHA256:
 CE:CD:DC:90:50:99:D8:DA:DF:C5:B1:D2:09:B7:37:CB:E2:C1:8C:FB:2C:10:C0:FF:0B:CF:0D:32:86:FC:1A:A
Alias name: xrampglobalcaroot
  SHA1: B8:01:86:D1:EB:9C:86:A5:41:04:CF:30:54:F3:4C:52:B7:E5:58:C6
  SHA256:
 CE:CD:DC:90:50:99:D8:DA:DF:C5:B1:D2:09:B7:37:CB:E2:C1:8C:FB:2C:10:C0:FF:0B:CF:0D:32:86:FC:1A:A
```

Utilizzalo AWS WAF per proteggere le tue API REST in API Gateway

AWS WAF è un firewall per applicazioni Web che aiuta a proteggere le applicazioni Web e le API dagli attacchi. Consente di configurare un set di regole denominato lista di controllo accessi Web (web ACL) per consentire, bloccare o contare le richieste Web in base a condizioni e regole di sicurezza Web personalizzabili definite dall'utente. Per ulteriori informazioni, consulta How AWS WAF Works.

Puoi utilizzarla AWS WAF per proteggere l'API REST di API Gateway da exploit web comuni, come attacchi SQL injection e cross-site scripting (XSS). Questi potrebbero influire sulla disponibilità e sulle prestazioni delle API, compromettere la sicurezza o consumare risorse eccessive. Ad esempio, puoi creare regole per consentire o bloccare richieste da intervalli di indirizzi IP specificati, richieste da blocchi CIDR, richieste provenienti da un paese o una regione specifici, richieste che contengono codice SQL dannoso o richieste contenenti script dannoso.

Puoi anche creare regole che corrispondano a una stringa specificata o a un modello di espressione regolare nelle intestazioni HTTP, nel metodo, nella stringa di query, nell'URI e nel corpo della richiesta (limitato ai primi 64 KB). Inoltre, puoi creare regole per bloccare attacchi da utenti-agenti, bad bot e scraper di contenuti. Ad esempio, puoi usare le regole basate sulla frequenza per specificare il numero di richieste Web consentite da ogni IP client in un periodo di 5 minuti, costantemente aggiornato, finale.

> **Important**
>
> AWS WAF è la tua prima linea di difesa contro gli exploit web. Quando AWS WAF è abilitato su un'API, AWS WAF le regole vengono valutate prima di altre funzionalità di controllo degli accessi, come le policy delle risorse, le policy IAM, gli autorizzatori Lambda e gli autorizzatori Amazon Cognito. Ad esempio, se AWS WAF blocca l'accesso da un blocco CIDR consentito da una politica delle risorse, ha la AWS WAF precedenza e la politica delle risorse non viene valutata.

AWS WAF Per abilitare la tua API, devi fare quanto segue:

1. Usa la AWS WAF console, l' AWS SDK o la CLI per creare un ACL Web che contenga la combinazione desiderata AWS WAF di regole gestite e regole personalizzate. Per ulteriori informazioni, consulta la sezione Guida introduttiva AWS WAF e elenchi di controllo degli accessi Web (ACL Web).

   > **Important**
   >
   > API Gateway richiede un ACL AWS WAFV2 web per un'applicazione regionale o un ACL AWS WAF Classic regionale web.

2. Associa l'ACL AWS WAF web a una fase API. Puoi farlo utilizzando la AWS WAF console, l' AWS SDK, la CLI o utilizzando la console API Gateway.

Per associare un ACL AWS WAF Web a uno stadio API Gateway API utilizzando la console API Gateway

Per utilizzare la console API Gateway per associare un ACL AWS WAF Web a uno stadio API Gateway API esistente, attenersi alla seguente procedura:

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Seleziona un'API esistente o creane una nuova.
3. Nel riquadro di navigazione principale scegli Fasi, quindi seleziona una fase.
4. Nella sezione Dettagli fase scegli Modifica.
5. In Web application firewall (AWS WAF), seleziona il tuo ACL web.

Se lo stai utilizzando AWS WAFV2, seleziona un ACL AWS WAFV2 web per un'applicazione regionale. L'ACL web e tutte AWS WAFV2 le altre risorse che utilizza devono trovarsi nella stessa regione dell'API.

Se lo stai utilizzando AWS WAF Classic regionale, seleziona un ACL web regionale.

6. Seleziona Salvataggio delle modifiche.

Associa un ACL AWS WAF Web a una fase API Gateway utilizzando il AWS CLI

Per utilizzare l' AWS CLI associazione di un ACL AWS WAFV2 Web per un'applicazione regionale a uno stage API Gateway API esistente, chiamate il associate-web-aclcomando, come nell'esempio seguente:

```
aws wafv2 associate-web-acl \
--web-acl-arn  arn:aws:wafv2:{region}:111122223333:regional/webacl/test-cli/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \
--resource-arn arn:aws:apigateway:{region}::/restapis/4wk1k4onj3/stages/prod
```

Per utilizzare la fase API Gateway AWS CLI per associare un ACL AWS WAF Classic regionale Web a una fase API Gateway API esistente, chiamate il associate-web-aclcomando, come nell'esempio seguente:

```
aws waf-regional associate-web-acl \
--web-acl-id 'aabc123a-fb4f-4fc6-becb-2b00831cadcf' \
--resource-arn 'arn:aws:apigateway:{region}::/restapis/4wk1k4onj3/stages/prod'
```

Associa un ACL AWS WAF web a uno stadio API utilizzando l' AWS WAF API REST

Per utilizzare l'API AWS WAFV2 REST per associare un ACL AWS WAFV2 Web per un'applicazione regionale a uno stadio API Gateway API esistente, utilizzate il comando AssociateWebACL, come nell'esempio seguente:

```
import boto3

wafv2 = boto3.client('wafv2')

wafv2.associate_web_acl(
    WebACLArn='arn:aws:wafv2:{region}:111122223333:regional/webacl/test/abc6aa3b-
fc33-4841-b3db-0ef3d3825b25',
```

```
    ResourceArn='arn:aws:apigateway:{region}::/restapis/4wk1k4onj3/stages/prod'
)
```

Per utilizzare l'API AWS WAF REST per associare un ACL AWS WAF Classic regionale Web a uno stadio API Gateway API esistente, utilizzate il comando AssociateWebACL, come nell'esempio seguente:

```
import boto3

waf = boto3.client('waf-regional')

waf.associate_web_acl(
    WebACLId='aabc123a-fb4f-4fc6-becb-2b00831cadcf',
    ResourceArn='arn:aws:apigateway:{region}::/restapis/4wk1k4onj3/stages/prod'
)
```

Limita le richieste alle tue API REST per una migliore velocità di trasmissione in API Gateway

È possibile configurare limitazioni e quote per le API per proteggerle da eventuali troppe richieste. Sia le limitazioni che le quote sono applicate sulla base del massimo sforzo e dovrebbero essere considerate come obbiettivi piuttosto che massimali garantiti delle richieste.

API Gateway limita la larghezza di banda della rete delle richieste nell'API mediante l'algoritmo di token bucket, dove un token rappresenta una richiesta. Nello specifico, API Gateway esamina il tasso e i picchi di invio di richieste per tutte le API dell'account, per regione. Nell'algoritmo di bucket token, un picco può consentire il superamento predefinito di tali limiti, ma anche altri fattori possono causare il superamento dei limiti in alcuni casi.

Quando le richieste inviate superano i limiti impostati per il tasso a regime e per i limiti di picco, API Gateway inizia a limitare le richieste. I clienti potrebbero ricevere risposte agli errori `429 Too Many Requests` a questo punto. Quando rileva queste eccezioni, il client può reinviare le richieste non riuscite in modo da limitare la velocità.

In qualità di sviluppatore dell'API, puoi impostare i limiti di destinazione per le singole fasi o i singoli metodi dell'API per migliorare le prestazioni generali in tutte le API dell'account. In alternativa, puoi abilitare i piani di utilizzo per impostare limitazioni sugli invii di richieste al client in base a quote e tassi di richiesta specificati.

Argomenti

- Come sono applicate le impostazioni del limite del throttling in API Gateway
- Limitazione a livello di account per regione
- Configurazione del throttling a livello di API e destinazioni delle limitazioni a livello di fase in un piano di utilizzo
- Configurazione delle destinazioni di limitazione della larghezza di banda della rete a livello di fase
- Configurazione delle destinazioni della limitazione a livello di metodo in un piano di utilizzo

Come sono applicate le impostazioni del limite del throttling in API Gateway

Prima di configurare le impostazioni throttle e quota per la tua API, è utile comprendere i tipi di impostazioni relative al throttling per la tua API e come API Gateway le applica.

Amazon API Gateway fornisce due tipi base di impostazioni correlate alle impostazioni delle limitazioni:

- AWS i limiti di limitazione vengono applicati a tutti gli account e i clienti in una regione. Queste impostazioni del limite servono a evitare di sovraccaricare l'API, e di conseguenza il tuo account, con troppe richieste. Questi limiti sono stabiliti AWS e non possono essere modificati da un cliente.
- I limiti per account vengono applicati a tutte le API di un account in una regione specificata. Il limite del tasso a livello di account può essere aumentato - sono possibili limiti più elevati con API con timeout più brevi e payload più piccoli. Per richiedere un aumento dei limiti di limitazione a livello di account per Regione, contatta il Centro assistenza AWS. Per ulteriori informazioni, consulta Quote e note importanti. Tieni presente che questi limiti non possono essere superiori ai limiti di AWS limitazione.
- I limiti di throttling per API e per fase vengono applicati a livello di metodo API per una fase. È possibile configurare le stesse impostazioni per tutti i metodi o configurare impostazioni di throttling diverse per ciascun metodo. Tieni presente che questi limiti non possono essere superiori ai limiti di AWS limitazione.
- Limiti di throttling per client vengono applicati ai client che utilizzano le chiavi API associate al piano di utilizzo come identificatore client. Ricorda che questi limiti non possono essere più alti dei limiti per account.

API Gateway applica le impostazioni relative al throttling nel seguente ordine:

1. Limiti di throttling per metodo o per client impostati per una fase API in un piano di utilizzo
2. Limiti di throttling per metodo impostati per una fase API

3. Limitazione a livello di account per regione
4. AWS Limitazione regionale

Limitazione a livello di account per regione

Per impostazione predefinita, API Gateway limita il numero di richieste con stato costante al secondo (RPS) per tutte le API all'interno di un account AWS , per Regione. Limita anche i picchi, ovvero la dimensione massima del bucket, per tutte le API di un account AWS , per regione. In API Gateway, il limite dei picchi rappresenta il numero massimo di invii di richieste che API Gateway può gestire prima di restituire risposte di errore `429 Too Many Requests`. Per ulteriori informazioni sulla limitazione delle quote, vedere Quote e note importanti.

Configurazione del throttling a livello di API e destinazioni delle limitazioni a livello di fase in un piano di utilizzo

In un piano di utilizzo puoi impostare una destinazione della limitazione (della larghezza di banda della rete) per tutti i metodi a livello di API o di fase. È possibile specificare una velocità di limitazione (della larghezza di banda della rete)ovvero la frequenza, in richieste al secondo, con cui i token vengono aggiunti al bucket di token. È anche possibile specificare un burst di limitazione (della larghezza di banda della rete), ovvero la capacità del bucket di token.

Puoi utilizzare la AWS CLI, gli SDK e AWS Management Console per creare un piano di utilizzo. Per ulteriori informazioni su come creare un piano di utilizzo, consulta. ???

Configurazione delle destinazioni di limitazione della larghezza di banda della rete a livello di fase

Puoi utilizzare la AWS CLI, gli SDK e il AWS Management Console per creare obiettivi di throttling a livello di stage.

Per ulteriori informazioni su come utilizzare per creare obiettivi di limitazione a livello di fase, AWS Management Console consulta. ??? Per ulteriori informazioni su come utilizzare la AWS CLI per creare obiettivi di throttling a livello di fase, consulta create-stage.

Configurazione delle destinazioni della limitazione a livello di metodo in un piano di utilizzo

Puoi impostare destinazioni della limitazione aggiuntive a livello di metodo in Usage Plans (Piani di utilizzo) come mostrato in Creazione di un piano di utilizzo. Nella console API Gateway, questi

vengono impostati specificando `Resource=<resource>`, `Method=<method>` nell'impostazione Configure Method Throttling (Configurazione del throttling). Ad esempio, ad esempio, è possibile specificare,. PetStore `Resource=/pets` `Method=GET`

Privato REST APIs in API Gateway

Un privato API è un REST API messaggio richiamabile solo dall'interno di AmazonVPC. Puoi accedere al tuo API utilizzando un VPCendpoint di interfaccia, che è un'interfaccia di rete di endpoint che crei nel tuo. VPC Endpoint di interfaccia con tecnologia AWS PrivateLink, una tecnologia che consente di accedere in modo privato ai AWS servizi utilizzando indirizzi IP privati.

Puoi anche utilizzare AWS Direct Connect per stabilire una connessione da una rete locale ad Amazon VPC e quindi accedere alla tua rete privata API tramite quella connessione. In tutti i casi, il traffico verso la rete privata API utilizza connessioni sicure ed è isolato dalla rete Internet pubblica. Il traffico non esce dalla rete Amazon.

Le migliori pratiche per i privati APIs

Ti consigliamo di utilizzare le seguenti best practice quando crei il tuo account privatoAPI.

- Utilizza un singolo VPC endpoint per accedere a più endpoint privatiAPIs. Ciò riduce il numero di VPC endpoint di cui potresti aver bisogno.
- Associa il tuo VPC endpoint al tuo. API Questo crea un DNS record di alias Route 53 e semplifica l'invocazione del tuo account privato. API
- Attiva la modalità privata DNS per te. VPC Quando attivi la modalità privata DNS per teVPC, puoi richiamare il tuo indirizzo API interno a VPC senza passare l'`x-apigw-api-id`intestazione `Host` o.

  Se attivi la modalità privataDNS, non puoi accedere all'endpoint predefinito per il pubblico. APIs Per accedere all'endpoint pubblico predefinitoAPIs, ti consigliamo di creare una zona ospitata privata per ogni dispositivo privato API dell'utente VPC e quindi di fornire i record richiesti in Route 53. Ciò consente all'endpoint privato API di risolversi mentre è ancora possibile richiamare l'endpoint pubblico predefinito dal proprio. VPC Per ulteriori informazioni consulta Creating a private hosted zone (Creazione di una zona ospitata privata).
- Limita l'accesso al tuo account privato API a endpoint specifici VPCs o VPC specifici. Aggiungi `aws:SourceVpc` o `aws:SourceVpce` condiziona la tua API politica sulle risorse per limitare l'accesso.

- Per il perimetro dei dati più sicuro, puoi creare una policy per gli VPC endpoint. Questo controlla l'accesso agli VPC endpoint che possono invocare il tuo account privato. API

Considerazioni per uso privato APIs

Le seguenti considerazioni potrebbero influire sull'utilizzo del privato. APIs

- Solo REST APIs sono supportati.
- I nomi di dominio personalizzati non sono supportati per uso privatoAPIs.
- Non è possibile convertire un file privato API in uno ottimizzato per i bordiAPI.
- Supporto APIs solo privato 1.2. TLS TLSLe versioni precedenti non sono supportate.
- VPCgli endpoint for private APIs sono soggetti alle stesse limitazioni degli altri VPC endpoint di interfaccia. Per ulteriori informazioni, consulta Accedere a un AWS servizio utilizzando un VPC endpoint di interfaccia nella Guida.AWS PrivateLink Per ulteriori informazioni sull'utilizzo di API Gateway con sottoreti condivise VPCs e condivise, vedere Subnet condivise nella Guida.AWS PrivateLink

Passaggi successivi per il settore privato APIs

Per sapere come creare un VPC endpoint privato API e associarlo, vedi,the section called “Crea un account privato API”. Per seguire un tutorial in cui si creano dipendenze in AWS CloudFormation e una privata API in AWS Management Console, vedi. the section called “Tutorial: crea un account privato REST API”

Crea un account privato API

Prima di creare un endpoint privatoAPI, devi prima creare un VPC endpoint per API Gateway. Quindi crei il tuo account privato API e gli alleghi una politica delle risorse. Facoltativamente, puoi associare il tuo VPC endpoint a quello privato API per semplificare il modo in cui richiami il tuo. API Infine, distribuisci il tuo. API

Le seguenti procedure descrivono come eseguire questa operazione. È possibile creare un file privato REST API utilizzando AWS Management Console, AWS CLI o un AWS SDK.

Prerequisiti

Per seguire questi passaggi, è necessario disporre di un file completamente configuratoVPC. Per informazioni su come creare un fileVPC, consulta Create a VPC only nella Amazon VPC User Guide.

Per seguire tutti i passaggi consigliati durante la creazione del tuoVPC, abilita privateDNS. In questo modo puoi invocare your API within a VPC senza dover passare l'Host o l'`x-apigw-api-id`header.

Per abilitare privateDNS, gli `enableDnsHostnames` attributi `enableDnsSupport` and di your VPC devono essere impostati su. `true` Per ulteriori informazioni, consulta DNSSupport in Your VPC e Updating DNS Support for Your VPC.

Passaggio 1: crea un VPC endpoint per API Gateway nel tuo VPC

La procedura seguente mostra come creare un VPC endpoint per API Gateway. Per creare un VPC endpoint per API Gateway, specificate il `execute-api` dominio Regione AWS in cui create il vostro dispositivo privato. API Il `execute-api` dominio è il servizio componente API Gateway per API l'esecuzione.

Quando crei l'VPCendpoint per API Gateway, specifichi le DNS impostazioni. Se disattivi la modalità privataDNS, puoi accedere solo API tramite pubblicoDNS. Per ulteriori informazioni, consulta the section called “Problema: non riesco a connettermi al mio dispositivo pubblico API da un VPC endpoint API Gateway”.

AWS Management Console

Per creare un VPC endpoint di interfaccia per API Gateway

1. Accedi a AWS Management Console e apri la VPC console Amazon all'indirizzo https://console.aws.amazon.com/vpc/.
2. Dal riquadro di navigazione, in Cloud privato virtuale, scegli Endpoint.
3. Seleziona Crea endpoint.
4. (Facoltativo) Per il tag Name, inserisci un nome per identificare l'VPCendpoint.
5. Per Service category (Categoria servizio), scegli AWS services.
6. In Servizi, nella barra di ricerca, inserisci**`execute-api`**. Quindi, scegli l'endpoint del servizio API Gateway in Regione AWS cui creerai il tuoAPI. Il nome del servizio dovrebbe essere simile `com.amazonaws.us-east-1.execute-api` e il tipo dovrebbe essere Interface.
7. Per VPC, scegli VPC quello in cui vuoi creare l'endpoint.
8. (Facoltativo) Per disattivare Abilita DNS nome privato, scegli Impostazioni aggiuntive e quindi deseleziona Abilita DNS nome privato.
9. Per le sottoreti, scegli le zone di disponibilità in cui hai creato le interfacce di rete degli endpoint. Per migliorare la disponibilità delle tueAPI, scegli più sottoreti.

10. Per Gruppo di sicurezza, seleziona il gruppo di sicurezza da associare alle interfacce di rete degli VPC endpoint.

    Il gruppo di sicurezza scelto deve essere impostato in modo da consentire il HTTPS traffico in entrata sulla TCP Porta 443 proveniente da un intervallo IP del tuo VPC o da un altro gruppo di sicurezza del tuo. VPC

11. Per Policy, esegui una delle seguenti operazioni:

    - Se non hai creato una policy privata API o non desideri configurare una policy personalizzata per gli VPC endpoint, scegli Accesso completo.
    - Se hai già creato una policy privata API e desideri configurare una policy personalizzata per gli VPC endpoint, puoi inserire una policy personalizzata per gli VPC endpoint. Per ulteriori informazioni, consulta the section called “Usa le policy VPC degli endpoint per uso privato APIs”.

    È possibile aggiornare la policy degli VPC endpoint dopo aver creato l'endpoint. VPC Per ulteriori informazioni, consulta Aggiornare una policy per gli VPC endpoint.

12. Seleziona Crea endpoint.
13. Copia l'ID VPC endpoint risultante, poiché potresti utilizzarlo nelle fasi future.

AWS CLI

Il create-vpc-endpointcomando seguente può essere usato per creare un VPC endpoint:

```
aws ec2 create-vpc-endpoint \
    --vpc-id vpc-1a2b3c4d \
    --vpc-endpoint-type Interface \
    --service-name com.amazonaws.us-east-1.execute-api \
    --subnet-ids subnet-7b16de0c \
    --security-group-id sg-1a2b3c4d
```

Copia l'ID VPC endpoint risultante, poiché potresti utilizzarlo nelle fasi future.

Passaggio 2: crea un account privato API

Dopo aver creato il tuo VPC endpoint, ne crei uno privato RESTAPI. La procedura seguente mostra come creare un privatoAPI.

AWS Management Console

Per creare un ambiente privato API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli Crea. API
3. In RESTAPI, scegli Costruisci.
4. In Nome, immetti un nome.
5. (Facoltativo) In Description (Descrizione), immettere una descrizione.
6. Per il tipo di API endpoint, seleziona Privato.
7. (Facoltativo) Per VPCendpoint IDs, inserisci un ID VPC endpoint.

   Se associ un ID VPC endpoint al tuo ID privatoAPI, puoi richiamare il tuo modulo API dall'interno del tuo VPC senza sovrascrivere un'`Host`intestazione o passare un `x-apigw-api-id header` messaggio. Per ulteriori informazioni, consulta. the section called "(Facoltativo) Associa o dissocia un endpoint da un endpoint privato VPC API"
8. Scegli Crea. API

Dopo aver completato i passaggi precedenti, puoi seguire le istruzioni riportate in the section called "Inizia a usare la REST API console" per configurare metodi e integrazioni a tale scopoAPI, ma non puoi implementare il tuo. API Per implementare il tuoAPI, segui il passaggio 3 e allega una politica delle risorse al tuo. API

AWS CLI

Il create-rest-apicomando seguente mostra come creare un file privatoAPI:

```
aws apigateway create-rest-api \
        --name 'Simple PetStore (AWS CLI, Private)' \
        --description 'Simple private PetStore API' \
        --region us-west-2 \
        --endpoint-configuration '{ "types": ["PRIVATE"] }'
```

In caso di esito positivo, la chiamata restituisce un output simile al seguente:

```
{
    "createdDate": "2017-10-13T18:41:39Z",
    "description": "Simple private PetStore API",
```

```
    "endpointConfiguration": {
        "types": "PRIVATE"
    },
    "id": "0qzs2sy7bh",
    "name": "Simple PetStore (AWS CLI, Private)"
}
```

Dopo aver completato i passaggi precedenti, puoi seguire le istruzioni riportate in the section called “Tutorial: crea un'API REST utilizzando AWS SDK o AWS CLI” per configurare metodi e integrazioni a tale scopoAPI, ma non puoi implementare il tuo. API Per implementare il tuoAPI, segui il passaggio 3 e allega una politica delle risorse al tuo. API

SDK JavaScript v3

L'esempio seguente mostra come creare un file privato API utilizzando AWS SDK for JavaScript v3:

```
import {APIGatewayClient, CreateRestApiCommand} from "@aws-sdk/client-api-gateway";
const apig = new APIGatewayClient({region:"us-east-1"});

const input = { // CreateRestApiRequest
  name: "Simple PetStore (JavaScript v3 SDK, private)", // required
  description: "Demo private API created using the AWS SDK for JavaScript v3",
  version: "0.00.001",
  endpointConfiguration: { // EndpointConfiguration
    types: [ "PRIVATE"],
  },
};

export const handler = async (event) => {
const command = new CreateRestApiCommand(input);
try {
  const result = await apig.send(command);
  console.log(result);
} catch (err){
  console.error(err)
 }
};
```

In caso di esito positivo, la chiamata restituisce un output simile al seguente:

```
{
  apiKeySource: 'HEADER',
```

```
  createdDate: 2024-04-03T17:56:36.000Z,
  description: 'Demo private API created using the AWS SDK for JavaScript v3',
  disableExecuteApiEndpoint: false,
  endpointConfiguration: { types: [ 'PRIVATE' ] },
  id: 'abcd1234',
  name: 'Simple PetStore (JavaScript v3 SDK, private)',
  rootResourceId: 'efg567',
  version: '0.00.001'
}
```

Dopo aver completato i passaggi precedenti, puoi seguire le istruzioni riportate in the section called “Tutorial: crea un'API REST utilizzando AWS SDK o AWS CLI” per configurare metodi e integrazioni a tale scopoAPI, ma non puoi implementare il tuo. API Per implementare il tuoAPI, segui il passaggio 3 e allega una politica delle risorse al tuo. API

Python SDK

L'esempio seguente mostra come creare un file privato API usando AWS SDK for Python:

```
import json
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

def lambda_handler(event, context):
    try:
      result = apig.create_rest_api(
      name='Simple PetStore (Python SDK, private)',
      description='Demo private API created using the AWS SDK for Python',
      version='0.00.001',
      endpointConfiguration={
          'types': [
             'PRIVATE',
          ],
      },
      )
    except botocore.exceptions.ClientError as error:
            logger.exception("Couldn't create private API %s.", error)
            raise
    attribute=["id", "name", "description", "createdDate", "version",
 "apiKeySource", "endpointConfiguration"]
```

```
    filtered_data ={key:result[key] for key in attribute}
    result = json.dumps(filtered_data, default=str, sort_keys='true')
    return result
```

In caso di esito positivo, la chiamata restituisce un output simile al seguente:

```
"{\"apiKeySource\": \"HEADER\", \"createdDate\": \"2024-04-03 17:27:05+00:00\",
 \"description\": \"Demo private API created using the AWS SDK for \",
 \"endpointConfiguration\": {\"types\": [\"PRIVATE\"]}, \"id\": \"abcd1234\", \"name
\": \"Simple PetStore (Python SDK, private)\", \"version\": \"0.00.001\"}"
```

Dopo aver completato i passaggi precedenti, puoi seguire le istruzioni riportate in the section called “Tutorial: crea un'API REST utilizzando AWS SDK o AWS CLI” per configurare metodi e integrazioni a tale scopoAPI, ma non puoi implementare il tuo. API Per implementare il tuoAPI, segui il passaggio 3 e allega una politica delle risorse al tuo. API

Passaggio 3: Imposta una politica delle risorse per un privato API

Il tuo account privato attuale API è inaccessibile a tuttiVPCs. Utilizza una politica in materia di risorse per concedere a te VPCs e agli VPC endpoint l'accesso ai dati privati. APIs Puoi concedere l'accesso a un VPC endpoint in qualsiasi AWS account.

La tua politica sulle risorse dovrebbe contenere `aws:SourceVpc` o stabilire `aws:SourceVpce` condizioni per limitare l'accesso. Ti consigliamo di identificare VPC endpoint VPCs e dispositivi specifici e di non creare una politica delle risorse che consenta l'accesso a tutti VPCs gli VPC endpoint.

La procedura seguente mostra come allegare una politica delle risorse al tuo. API

AWS Management Console

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli un. REST API
3. Nel riquadro di navigazione principale, scegli Policy delle risorse.
4. Scegli Create Policy (Crea policy).
5. Scegli Seleziona un modello, quindi scegli Fonte VPC.
6. `{{vpceID}}`Sostituiscilo (comprese le bretelle arricciate) con il tuo ID VPC endpoint.
7. Scegli Save changes (Salva modifiche).

AWS CLI

Il update-rest-apicomando seguente mostra come allegare una politica delle risorse a una esistente: API

```
aws apigateway update-rest-api \
    --rest-api-id a1b2c3 \
    --patch-operations op=replace,path=/
policy,value='"{\"jsonEscapedPolicyDocument\"}"'
```

Potresti anche voler controllare quali risorse hanno accesso al tuo VPC endpoint. Per controllare quali risorse hanno accesso al tuo VPC endpoint, allega una policy sull'endpoint al tuo endpoint. VPC Per ulteriori informazioni, consulta the section called “Usa le policy VPC degli endpoint per uso privato APIs”.

(Facoltativo) Associa o dissocia un endpoint da un endpoint privato VPC API

Quando associ un VPC endpoint a un dispositivo privatoAPI, API Gateway genera un nuovo record di alias Route 53. DNS Puoi usare questo record per richiamare il tuo nome privato APIs proprio come fai con quello pubblico APIs senza sovrascrivere un'intestazione o passare un'`Host`intestazione. `x-apigw-api-id`

La base generata URL è nel seguente formato:

```
https://{rest-api-id}-{vpce-id}.execute-api.{region}.amazonaws.com/{stage}
```

Associate a VPC endpoint (AWS Management Console)

Puoi associare un VPC endpoint al tuo dispositivo privato al API momento della creazione o dopo la creazione. La procedura seguente mostra come associare un VPC endpoint a un dispositivo creato in precedenza. API

Per associare un VPC endpoint a un dispositivo privato API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli il tuo account privato. API
3. Nel riquadro di navigazione principale, scegli Policy delle risorse.
4. Modifica la politica delle risorse per consentire le chiamate dall'VPCendpoint aggiuntivo.

5. Nel riquadro di navigazione principale, scegli APIimpostazioni.
6. Nella sezione dei APIdettagli, scegli Modifica.
7. Per VPCendpoint IDs, seleziona VPC IDs endpoint aggiuntivo.
8. Seleziona Salva.
9. Ridistribuisci il tuo API per rendere effettive le modifiche.

Dissociate a VPC endpoint (AWS Management Console)

Per dissociare un VPC endpoint da uno privato REST API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com / apigateway.
2. Scegli il tuo account privato. API
3. Nel riquadro di navigazione principale, scegli Policy delle risorse.
4. Modifica la tua politica sulle risorse per rimuovere le menzioni dell'VPCendpoint che desideri separare da quelle private. API
5. Nel pannello di navigazione principale, scegli impostazioni. API
6. Nella sezione dei APIdettagli, scegli Modifica.
7. Per VPCendpoint IDs, scegli la X per dissociare l'endpoint. VPC
8. Seleziona Salva.
9. Ridistribuisci il tuo API per rendere effettive le modifiche.

Associate a VPC endpoint (AWS CLI)

Il create-rest-apicomando seguente mostra come associare gli VPC endpoint al momento della creazione: API

```
aws apigateway create-rest-api \
    --name Petstore \
    --endpoint-configuration '{ "types": ["PRIVATE"], "vpcEndpointIds" :
 ["vpce-0212a4ababd5b8c3e", "vpce-0393a628149c867ee"] }' \
    --region us-west-2
```

L'output sarà simile al seguente:

```
{
    "apiKeySource": "HEADER",
    "endpointConfiguration": {
        "types": [
            "PRIVATE"
        ],
        "vpcEndpointIds": [
            "vpce-0212a4ababd5b8c3e",
            "vpce-0393a628149c867ee"
        ]
    },
    "id": "u67n3ov968",
    "createdDate": 1565718256,
    "name": "Petstore"
}
```

Il update-rest-apicomando seguente mostra come associare gli VPC endpoint a un dispositivo API già creato:

```
aws apigateway update-rest-api \
    --rest-api-id u67n3ov968 \
    --patch-operations "op='add',path='/endpointConfiguration/
vpcEndpointIds',value='vpce-01d622316a7df47f9'" \
    --region us-west-2
```

L'output sarà simile al seguente:

```
{
    "name": "Petstore",
    "apiKeySource": "1565718256",
    "tags": {},
    "createdDate": 1565718256,
    "endpointConfiguration": {
        "vpcEndpointIds": [
            "vpce-0212a4ababd5b8c3e",
            "vpce-0393a628149c867ee",
            "vpce-01d622316a7df47f9"
        ],
        "types": [
            "PRIVATE"
        ]
```

```
    },
    "id": "u67n3ov968"
}
```

Ridistribuite il vostro API per rendere effettive le modifiche.

Disassociate a VPC endpoint (AWS CLI)

Il update-rest-apicomando seguente mostra come dissociare un VPC endpoint da un dispositivo privato: API

```
aws apigateway update-rest-api \
    --rest-api-id u67n3ov968 \
    --patch-operations "op='remove',path='/endpointConfiguration/
vpcEndpointIds',value='vpce-0393a628149c867ee'" \
    --region us-west-2
```

L'output sarà simile al seguente:

```
{
    "name": "Petstore",
    "apiKeySource": "1565718256",
    "tags": {},
    "createdDate": 1565718256,
    "endpointConfiguration": {
        "vpcEndpointIds": [
            "vpce-0212a4ababd5b8c3e",
            "vpce-01d622316a7df47f9"
        ],
        "types": [
            "PRIVATE"
        ]
    },
    "id": "u67n3ov968"
}
```

Ridistribuisci il tuo API per rendere effettive le modifiche.

Fase 4: Implementa un account privato API

Per implementare la tuaAPI, crei una API distribuzione e la associ a una fase. La procedura seguente mostra come implementare la versione privata. API

AWS Management Console

Per distribuire un dispositivo privato API

1. Scegli il tuo. API
2. Scegliere Deploy (Implementa)API.
3. In Fase, seleziona Nuova fase.
4. Per Nome fase immetti il nome di una fase.
5. (Facoltativo) In Description (Descrizione), immettere una descrizione.
6. Seleziona Deploy (Implementa).

AWS CLI

Il seguente comando create-deployment mostra come distribuire un file privato: API

```
aws apigateway create-deployment --rest-api-id a1b2c3 \
  --stage-name test \
  --stage-description 'Private API test stage' \
  --description 'First deployment'
```

Risolvi i problemi relativi al file privato API

Di seguito vengono forniti consigli per la risoluzione di errori e problemi che potresti riscontrare durante la creazione di un file privato. API

Problema: non riesco a connettermi al mio dispositivo pubblico API da un VPC endpoint API Gateway

Quando crei il tuoVPC, puoi configurare le DNS impostazioni. Ti consigliamo di attivare la modalità privata DNS per il tuoVPC. Se scegli di disattivare la modalità privataDNS, puoi accedere al tuo solo API tramite pubblicoDNS.

Se abiliti la modalità privataDNS, non puoi accedere all'endpoint predefinito di un API Gateway pubblico API dal tuo VPC endpoint. Puoi accedere a un dominio API con un nome di dominio personalizzato.

Se crei un nome di dominio personalizzato regionale, utilizzi un record di alias di tipo A, se crei un nome di dominio personalizzato ottimizzato per Edge, non ci sono restrizioni per il tipo di record. Puoi accedere a questi dati pubblici APIs con l'opzione privata abilitata. DNS Per ulteriori informazioni, vedi Problema: mi connetto al mio pubblico API da un VPC endpoint API Gateway.

Problema: i miei resi API **`{"Message":"User: anonymous is not authorized to perform: execute-api:Invoke on resource: arn:aws:execute-api:us-east-1:********/****/****/"}`**

Nella tua politica delle risorse, se imposti il AWS Principal su un principale, come segue:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            ""Principal": {
                "AWS": [
                    "arn:aws:iam::account-id:role/developer",
                    "arn:aws:iam::account-id:role/Admin"
                ]
            },
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        ...
        }
```

È necessario utilizzare `AWS_IAM` l'autorizzazione per ogni metodo utilizzatoAPI, altrimenti verrà API restituito il messaggio di errore precedente. Per ulteriori istruzioni su come attivare `AWS_IAM` l'autorizzazione per un metodo, consultathe section called “Metodi”.

Problema: non riesco a capire se il mio VPC endpoint è associato al mio API

Se associ o dissocii un VPC endpoint dal tuo dispositivo privatoAPI, devi ridistribuire il tuo. API Il completamento dell'operazione di aggiornamento potrebbe richiedere alcuni minuti a causa della propagazione. DNS Durante questo periodo, il tuo API è disponibile, ma la DNS propagazione della nuova generazione DNS URLs potrebbe essere ancora in corso. Se dopo alcuni minuti, i nuovi file non URLs si risolvonoDNS, si consiglia di ridistribuire il file. API

Invoca un privato API

È possibile invocare un endpoint privato solo API dall'interno di un VPC VPC endpoint. Il tuo utente privato API deve disporre di una politica delle risorse che consenta a specifici VPC endpoint VPCs di richiamare il tuo. API

Puoi invocare il tuo account privato API nei seguenti modi.

- Invoca il tuo API utilizzando un alias Route53. Questo è disponibile solo se hai associato il tuo endpoint al tuoVPC. API Per ulteriori informazioni, consulta the section called “(Facoltativo) Associa o dissocia un endpoint da un endpoint privato VPC API”.
- Invoca il tuo API utilizzo privato. DNS Questo è disponibile solo se hai abilitato private DNS per il tuoVPC.
- Invoca il tuo API utilizzo AWS Direct Connect.
- Invoca il tuo API utilizzando nomi di host pubblici specifici dell'endpoint. DNS

Per richiamare il tuo nome privato API utilizzando un DNS nome, devi identificare i nomi del tuo. DNS API La procedura seguente mostra come trovare i propri DNS nomi.

AWS Management Console

Per trovare i DNS nomi

1. Accedi a AWS Management Console e apri la VPC console Amazon all'indirizzo https://console.aws.amazon.com/vpc/.
2. Nel pannello di navigazione principale, scegli Endpoints, quindi scegli l'VPCendpoint di interfaccia per API Gateway.
3. Nel riquadro Dettagli, vedrai cinque valori nel campo dei DNSnomi. I primi tre sono i tuoi DNS nomi pubbliciAPI. Gli altri due sono i DNS suoi nomi privati.

AWS CLI

Usa il seguente describe-vpc-endpointscomando per elencare DNS i tuoi valori.

```
aws ec2 describe-vpc-endpoints --vpc-endpoint-ids vpce-01234567abcdef012
```

I primi tre sono i tuoi DNS nomi pubbliciAPI. Gli altri due sono i DNS suoi nomi privati.

Invoca un privato API utilizzando un alias Route53

Puoi associare o dissociare un endpoint dal tuo dispositivo privatoVPC. API Per ulteriori informazioni, consulta the section called “(Facoltativo) Associa o dissocia un endpoint da un endpoint privato VPC API”.

Dopo aver associato i tuoi VPC endpoint al tuo dispositivo privatoAPI, puoi utilizzare la seguente base URL per richiamare: API

```
https://{rest-api-id}-{vpce-id}.execute-api.{region}.amazonaws.com/{stage}
```

Ad esempio, se hai impostato il `GET /pets` metodo per lo `test` stage e il tuo REST API ID era`01234567ab`, il tuo ID VPC endpoint era e la tua regione era `vpce-01234567abcdef012us-west-2`, puoi invocare il tuo annuncio: API

```
curl -v https://01234567ab-vpce-01234567abcdef012.execute-api.us-west-2.amazonaws.com/
test/pets
```

Invoca un privato usando nomi privati API DNS

Se hai abilitato il privatoDNS, puoi accedere al tuo account privato API utilizzando il seguente DNS nome privato:

```
{restapi-id}.execute-api.{region}.amazonaws.com
```

La base URL per richiamarla API è nel seguente formato:

```
https://{restapi-id}.execute-api.{region}.amazonaws.com/{stage}
```

Ad esempio, se hai impostato il `GET /pets` metodo per lo `test` stage e il tuo REST API ID era `01234567ab` e la tua regione era`us-west-2`, puoi richiamare il valore privato API inserendo quanto segue URL in un browser:

```
https://01234567ab.execute-api.us-west-2.amazonaws.com/test/pets
```

In alternativa, puoi usare il seguente URL comando c per richiamare il tuo account privato: API

```
curl -X GET https://01234567ab.execute-api.us-west-2.amazonaws.com/test/pets
```

> **Warning**
>
> Se abiliti private DNS per il tuo VPC endpoint, non sarai in grado di accedere all'endpoint predefinito per il pubblico. APIs Per ulteriori informazioni, vedi Perché non riesco a connettermi al mio pubblico API da un endpoint API Gateway VPC? .

Invoca un utilizzo privato API AWS Direct Connect

Puoi usarlo AWS Direct Connect per stabilire una connessione privata dedicata da una rete locale ad Amazon VPC e accedere al tuo API endpoint privato tramite tale connessione utilizzando nomi pubbliciDNS.

Puoi anche usare DNS nomi privati per accedere ai dati privati API da una rete locale configurando un endpoint in Amazon Route 53 Resolver entrata e inoltrando tutte le DNS richieste del privato dalla tua rete remota. DNS Per ulteriori informazioni, consulta Inoltro DNS delle query in entrata al tuo indirizzo nella Amazon Route 53 VPCs Developer Guide.

Invoca un server privato utilizzando nomi di host pubblici specifici dell'endpoint API DNS

Puoi accedere al tuo account privato utilizzando nomi host specifici dell'endpoint. API DNS Si tratta di DNS nomi di host pubblici contenenti l'ID dell'VPCendpoint o l'ID privato. API API

La base generata URL è nel seguente formato:

```
https://{public-dns-hostname}.execute-api.{region}.vpce.amazonaws.com/{stage}
```

Ad esempio, se impostate il `GET /pets` metodo per lo `test` stage e il vostro REST API ID era`abc1234`, il suo DNS hostname pubblico era `vpce-def-01234567` e la vostra regione sì`us-west-2`, potreste richiamare il nome privato API utilizzando il relativo VPCe ID utilizzando l'`Host`intestazione in un comando c: URL

```
curl -v https://vpce-def-01234567.execute-api.us-west-2.vpce.amazonaws.com/test/pets -H
 'Host: abc1234.execute-api.us-west-2.amazonaws.com'
```

In alternativa, puoi richiamare il tuo nome privato API tramite il relativo API ID utilizzando l'`x-apigw-api-id`intestazione in un URL comando c nel seguente formato:

```
curl -v https://{public-dns-hostname}.execute-api.{region}.vpce.amazonaws.com/{stage} -
H 'x-apigw-api-id:{api-id}'
```

Monitora le API REST in API Gateway

In questa sezione, puoi imparare a monitorare la tua API utilizzando CloudWatch metrics, CloudWatch Logs, Firehose e. AWS X-Ray Combinando log di CloudWatch esecuzione e

CloudWatch metriche, puoi registrare errori e tracce di esecuzione e monitorare le prestazioni dell'API. Potresti anche voler registrare le chiamate API su Firehose. Puoi anche utilizzarlo AWS X-Ray per tracciare le chiamate tramite i servizi downstream che compongono la tua API.

> **Note**
>
> L'API Gateway potrebbe non generare log e parametri nei seguenti casi:
>
> - Errori 413 per richiesta troppo grande
> - Errori 429 per troppe richieste
> - Errori serie 400 per richieste inviate a un dominio personalizzato che non dispone del mapping API
> - Errori serie 500 per malfunzionamenti interni
>
> API Gateway non genererà log e parametri durante il test di un metodo REST API. Le CloudWatch immissioni sono simulate. Per ulteriori informazioni, consulta the section called "Utilizzo della console per il test di un metodo API REST.".

Argomenti

- Monitora REST API l'esecuzione con i CloudWatch parametri di Amazon
- Configurare la CloudWatch registrazione per REST APIs Gateway API
- Registra le chiamate API REST su Amazon Data Firehose in API Gateway
- Traccia le richieste degli utenti sulle API REST utilizzando X-Ray in API Gateway

Monitora REST API l'esecuzione con i CloudWatch parametri di Amazon

È possibile monitorare API l'esecuzione utilizzando CloudWatch, che raccoglie ed elabora i dati grezzi da API Gateway in metriche leggibili. near-real-time Queste statistiche vengono registrate per un periodo di 15 mesi, per permettere l'accesso alle informazioni storiche e offrire una prospettiva migliore sulle prestazioni del servizio o dell'applicazione Web. Per impostazione predefinita, i dati metrici di API Gateway vengono inviati automaticamente in periodi di un CloudWatch minuto. Per ulteriori informazioni, consulta What Is Amazon CloudWatch? nella Amazon CloudWatch User Guide.

Le metriche riportate da API Gateway forniscono informazioni che puoi analizzare in diversi modi. L'elenco seguente mostra alcuni usi comuni dei parametri che sono suggerimenti per iniziare:

- Monitora le IntegrationLatencymetriche per misurare la reattività del backend.
- Monitora le metriche di latenza per misurare la reattività complessiva delle chiamate. API
- Monitora le CacheMissCountmetriche CacheHitCounte per ottimizzare le capacità della cache e ottenere le prestazioni desiderate.

Argomenti

- Dimensioni e parametri di Amazon API Gateway
- Visualizza le CloudWatch metriche con la API dashboard in API Gateway
- Visualizza le metriche del API gateway nella console CloudWatch
- Visualizza gli eventi di log del API Gateway nella console CloudWatch
- Strumenti di monitoraggio integrati per AWS Gateway API

Dimensioni e parametri di Amazon API Gateway

Le metriche e le dimensioni che API Gateway invia ad Amazon CloudWatch sono elencate di seguito. Per ulteriori informazioni, consulta Monitora REST API l'esecuzione con i CloudWatch parametri di Amazon.

Parametri di API Gateway

Amazon API Gateway invia dati metrici CloudWatch ogni minuto.

Lo spazio dei nomi `AWS/ApiGateway` include le metriche descritte di seguito.

Metrica	Descrizione
4XXError	Il numero di errori lato client catturati in un dato periodo. API Gateway considera i codici di stato di risposta del gateway modificati come errori 4xxError. La statistica `Sum` rappresenta questo parametro , ovvero il numero totale di errori 4XXError nel dato periodo. La statistica `Average` rappresenta la percentuale di errori 4XXError nel dato periodo, ovvero il numero totale di errori 4XXError diviso

Metrica	Descrizione
	per il numero totale di richieste durante il periodo. Il denominatore corrisponde al parametro Count (di seguito). Unit: Count
5XXError	Il numero di errori lato server catturati in un dato periodo. La statistica `Sum` rappresenta questo parametro , ovvero il numero totale di errori 5XXError nel dato periodo. La statistica `Average` rappresenta la percentuale di errori 5XXError nel dato periodo, ovvero il numero totale di errori 5XXError diviso per il numero totale di richieste durante il periodo. Il denominatore corrisponde al parametro Count (di seguito). Unit: Count
CacheHitCount	Il numero di richieste servite dalla cache API in un dato periodo. La statistica `Sum` rappresenta questo parametro, ovvero il numero totale di riscontri nella cache nel dato periodo. La statistica `Average` rappresenta la percentuale di riscontri nella cache, ovvero il numero totale di riscontri nella cache diviso per il numero totale di richieste durante il periodo. Il denominatore corrisponde al parametro Count (di seguito). Unit: Count

Metrica	Descrizione
CacheMissCount	Il numero di richieste servite dal back-end in un dato periodo, quando la memorizzazione nella cache delle API è abilitata. La statistica `Sum` rappresenta questo parametro , ovvero il numero totale di mancati riscontri nella cache nel dato periodo. La statistica `Average` rappresenta la percentuale di mancati riscontri nella cache, ovvero il numero totale di mancati riscontri nella cache diviso per il numero totale di richieste durante il periodo. Il denominatore corrisponde al parametro Count (di seguito). Unit: Count
Count	Il numero totale di richieste API in un dato periodo. La statistica `SampleCount` rappresenta questo parametro. Unit: Count
IntegrationLatency	Il tempo che intercorre tra il momento in cui API Gateway ritrasmette una richiesta al back-end e quando riceve da questo una risposta. Unit: Millisecond
Latency	Il tempo che intercorre tra il momento in cui API Gateway riceve una richiesta dal client e quando restituisce ad esso una risposta. La latenza include la latenza di integrazione e altri sovraccarichi dell'API Gateway. Unit: Millisecond

Dimensioni per i parametri

Per filtrare i parametri di API Gateway, puoi utilizzare le dimensioni nella seguente tabella.

> **Note**
>
> API Gateway rimuove i caratteri non ASCII dalla ApiName dimensione prima di inviare le metriche a. CloudWatch Se APIName non contiene caratteri ASCII, API ID viene utilizzato come ApiName.

Dimensione	Descrizione
ApiName	Filtra i parametri dell'API Gateway per l'API REST con il nome API specificato.
ApiName, Method, Resource, Stage	Filtra i parametri dell'API Gateway per il metodo API con nome API, fase, risorsa e metodo specificati. API Gateway non invierà queste metriche a meno che tu non abbia abilitato esplicitamente le metriche dettagliate CloudWatch . Nella console, scegli una fase, quindi seleziona Modifica per Log e tracciame nto. Seleziona Parametri dettagliati, quindi scegli Salva modifiche. In alternativa, puoi chiamare il AWS CLI comando update-stage per aggiornare la proprietà a. `metricsEnabled true` L'abilitazione di questi parametri comporta addebiti aggiuntivi sul tuo account. Per informazioni sui prezzi, consulta la pagina CloudWatchdei prezzi di Amazon.
ApiName, Stage	Filtra i parametri di API Gateway per la risorsa della fase API con il nome e la fase specificati per l'API.

Visualizza le CloudWatch metriche con la API dashboard in API Gateway

Puoi utilizzare la API dashboard nella console API Gateway per visualizzare le CloudWatch metriche del tuo dispositivo distribuito API in Gateway. API Queste vengono mostrate come un riepilogo dell'APIattività nel tempo.

Argomenti

- Prerequisiti
- Esamina API le attività nella dashboard

Prerequisiti

1. È necessario averne API creato uno in API Gateway. Segui le istruzioni in Sviluppa REST APIs in API Gateway.
2. È necessario averlo API distribuito almeno una volta. Segui le istruzioni in Implementazione REST APIs in Gateway API.

Esamina API le attività nella dashboard

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere un API.
3. Nel pannello di navigazione principale, seleziona Pannello di controllo.
4. In Fase, scegli la fase desiderata.
5. Scegli Intervallo di date per specificare un intervallo di date.
6. Aggiorna, se necessario, e visualizza le singole metriche visualizzate in grafici separati intitolati APIchiamate, latenza, latenza di integrazione, latenza, errore 4xx ed errore 5xx.

> **Tip**
>
> Per esaminare le CloudWatch metriche a livello di metodo, assicurati di aver abilitato i log a livello di metodo. CloudWatch Per ulteriori informazioni su come configurare la registrazione dei log a livello di metodo, consultare Sostituzione delle impostazioni a livello di fase.

Visualizza le metriche del API gateway nella console CloudWatch

I parametri vengono raggruppati prima in base allo spazio dei nomi del servizio e successivamente in base alle diverse combinazioni di dimensioni all'interno di ogni spazio dei nomi. Per visualizzare le metriche a livello di metodo per il tuoAPI, attiva le metriche dettagliate. Per ulteriori informazioni, consulta the section called “Aggiornamento delle impostazioni della fase”.

Per visualizzare le metriche del API Gateway utilizzando la console CloudWatch

1. Apri la CloudWatch console all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Se necessario, modifica Regione AWS. Dalla barra di navigazione, seleziona la regione in cui risiedono AWS le tue risorse.
3. Nel riquadro di navigazione, seleziona Parametri.
4. Nella scheda Tutte le metriche, scegli API Gateway.
5. Per visualizzare i parametri in base alla fase, selezionare il pannello By Stage (Per fase). Quindi, seleziona il tuo nome APIs e quello delle metriche.
6. Per visualizzare le metriche in base a uno specificoAPI, scegli il pannello By Api Name. Quindi, seleziona il tuo nome APIs e quello delle metriche.

Per visualizzare le metriche utilizzando il AWS CLI

1. Al prompt dei comandi utilizza il comando seguente per elencare i parametri:

   ```
   aws cloudwatch list-metrics --namespace "AWS/ApiGateway"
   ```

   Dopo aver creato una metrica, attendi fino a 15 minuti per la visualizzazione della metrica. Per visualizzare prima le statistiche delle metriche, usa o. get-metric-dataget-metric-statistics
2. Per visualizzare statistiche specifiche (ad esempio `Average`) per un periodo di tempo di intervalli di 5 minuti, chiama il seguente comando:

   ```
   aws cloudwatch get-metric-statistics --namespace AWS/ApiGateway --metric-name Count
    --start-time 2011-10-03T23:00:00Z --end-time 2017-10-05T23:00:00Z --period 300 --
   statistics Average
   ```

Visualizza gli eventi di log del API Gateway nella console CloudWatch

La sezione seguente spiega i prerequisiti necessari e come visualizzare gli eventi di registro del API Gateway nella CloudWatch console.

Prerequisiti

1. È necessario averne API creato uno in API Gateway. Segui le istruzioni in Sviluppa REST APIs in API Gateway.
2. È necessario che sia API distribuito e richiamato almeno una volta. A tale scopo, segui le istruzioni riportate in Implementazione REST APIs in Gateway API e Richiama le API REST in API Gateway.
3. È necessario che i CloudWatch registri siano abilitati per una fase. Segui le istruzioni in Configurare la CloudWatch registrazione per REST APIs Gateway API.

Per visualizzare le API richieste e le risposte registrate utilizzando la console CloudWatch

1. Apri la CloudWatch console all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Se necessario, modifica Regione AWS. Dalla barra di navigazione, seleziona la regione in cui risiedono AWS le tue risorse. Per ulteriori informazioni, consulta Regioni ed endpoint di .
3. Nel pannello di navigazione a sinistra, scegli Log, Gruppi di log.
4. Nella tabella Log Groups, scegli un gruppo di log del nome API-Gateway-Execution-Logs_ {}/{stage-name}. rest-api-id
5. Nella tabella Log Streams (Flussi di log), scegliere un flusso di log. Puoi utilizzare il timestamp per individuare il flusso di log che ti interessa.
6. Selezionare Text (Testo) per vedere il testo non elaborato o scegliere Row (Riga) per visualizzare l'evento riga dopo riga.

> **Important**
>
> CloudWatch consente di eliminare gruppi o stream di log. Non eliminate manualmente i gruppi o i flussi di API log del API Gateway; lasciate che API Gateway gestisca queste risorse. L'eliminazione manuale dei gruppi o dei flussi di log può comportare la mancata registrazione delle API richieste e delle risposte. In tal caso, è possibile eliminare l'intero

gruppo di log API e ridistribuire il. API Questo perché API Gateway crea gruppi di log o flussi di log per una API fase nel momento in cui viene distribuito.

Strumenti di monitoraggio integrati per AWS Gateway API

AWS fornisce vari strumenti che è possibile utilizzare per monitorare API Gateway. Alcuni di questi strumenti possono essere configurati in modo che eseguano automaticamente il monitoraggio, mentre altri richiedono l'intervento manuale. Si consiglia di automatizzare il più possibile le attività di monitoraggio.

Strumenti di monitoraggio automatizzati in AWS

Puoi utilizzare i seguenti strumenti di monitoraggio automatizzato per guardare API Gateway e segnalare quando qualcosa non va:

- Amazon CloudWatch Alarms: monitora una singola metrica in un periodo di tempo specificato ed esegui una o più azioni in base al valore della metrica rispetto a una determinata soglia in diversi periodi di tempo. L'azione è una notifica inviata a un argomento di Amazon Simple Notification Service (AmazonSNS) o a una politica di Amazon EC2 Auto Scaling. CloudWatch gli allarmi non richiamano azioni semplicemente perché si trovano in uno stato particolare; lo stato deve essere cambiato e mantenuto per un determinato numero di periodi. Per ulteriori informazioni, consulta Monitora REST API l'esecuzione con i CloudWatch parametri di Amazon.
- Amazon CloudWatch Logs: monitora, archivia e accedi ai tuoi file di registro da AWS CloudTrail o altre fonti. Per ulteriori informazioni, consulta What is CloudWatch Logs? nella Amazon CloudWatch User Guide.
- Amazon EventBridge (precedentemente chiamato CloudWatch Events): abbina gli eventi e li indirizza a una o più funzioni o flussi di destinazione per apportare modifiche, acquisire informazioni sullo stato e intraprendere azioni correttive. Per ulteriori informazioni, consulta What Is Amazon EventBridge? nella Guida EventBridge per l'utente.
- AWS CloudTrail Monitoraggio dei registri: condividi i file di CloudTrail registro tra account, monitora i file di registro in tempo reale inviandoli a CloudWatch Logs, scrivi applicazioni di elaborazione dei log in Java e verifica che i file di registro non siano stati modificati dopo la consegna da parte di. CloudTrail Per ulteriori informazioni, consulta Lavorare con i file di CloudTrail registro nella Guida per l'AWS CloudTrail utente.

Strumenti di monitoraggio manuali

Un'altra parte importante del monitoraggio di API Gateway consiste nel monitorare manualmente gli elementi non coperti dagli CloudWatch allarmi. Il API Gateway e le altre dashboard della AWS console forniscono una at-a-glance panoramica dello stato dell'ambiente AWS . CloudWatch Si consiglia di controllare anche i file di registro durante API l'esecuzione.

- APILa dashboard di Gateway mostra le seguenti statistiche per una determinata API fase durante un periodo di tempo specificato:
  - APIChiamate
  - Cache Hit, solo quando la API memorizzazione nella cache è abilitata.
  - Cache Miss, solo quando la API memorizzazione nella cache è abilitata.
  - Latenza
  - Integration Latency (Latenza di integrazione)
  - 4XX Error (Errore 4XX)
  - 5XX Error (Errore 5XX)
- La CloudWatch home page mostra:
  - Stato e allarmi attuali
  - Grafici degli allarmi e delle risorse
  - Stato di integrità dei servizi

  Inoltre, è possibile utilizzare CloudWatch per effettuare le seguenti operazioni:
  - Crea pannelli di controllo personalizzati per monitorare i servizi di interesse.
  - Crea grafici dei dati dei parametri per la risoluzione di problemi e il rilevamento di tendenze.
  - Cerca e sfoglia tutte le metriche AWS delle tue risorse
  - Crea e modifica gli allarmi per ricevere le notifiche dei problemi.

Creazione di CloudWatch allarmi per monitorare Gateway API

Puoi creare un CloudWatch allarme che invia un SNS messaggio Amazon quando l'allarme cambia stato. Un allarme controlla un singolo parametro in un periodo di tempo specificato ed esegue una o più operazioni in base al valore del parametro relativo a una determinata soglia in una serie di periodi di tempo. L'azione è una notifica inviata a un SNS argomento di Amazon o a una politica di Auto Scaling. Gli allarmi richiamano azioni solo per modifiche di stato sostenute. CloudWatch gli allarmi

non richiamano azioni semplicemente perché si trovano in uno stato particolare; lo stato deve essere cambiato e mantenuto per un determinato numero di periodi.

Configurare la CloudWatch registrazione per REST APIs Gateway API

Per risolvere i problemi relativi all'esecuzione della richiesta o all'accesso del client al tuoAPI, puoi abilitare Amazon CloudWatch Logs per registrare API le chiamate. Per ulteriori informazioni su CloudWatch, consulta. the section called “CloudWatch metriche”

CloudWatch formati di registro per API Gateway

Esistono due tipi di API registrazione CloudWatch: registrazione dell'esecuzione e registrazione degli accessi. Nella registrazione dell'esecuzione, API Gateway gestisce i registri. CloudWatch Il processo include la creazione di gruppi e flussi di log e la segnalazione ai flussi di log delle richieste e delle risposte dell'intermediario.

I dati registrati includono errori o tracce di esecuzione (come valori dei parametri di richiesta o risposta o payload), dati utilizzati dagli autorizzatori Lambda (precedentemente noti come autorizzatori personalizzati), se sono necessarie API le chiavi, se i piani di utilizzo sono abilitati e altre informazioni. APIGateway rimuove le intestazioni di autorizzazione, i valori delle API chiavi e parametri di richiesta sensibili simili dai dati registrati.

Quando si implementa unAPI, API Gateway crea un gruppo di log e dei flussi di log all'interno del gruppo di log. Al gruppo di log viene assegnato un nome nel formato `API-Gateway-Execution-Logs_{rest-api-id}/{stage_name}`. All'interno di ciascun gruppo, i log sono suddivisi ulteriormente in flussi, che vengono ordinati in base al valore Last Event Time (Ora ultimo evento) quando vengono riportati i dati registrati.

Nella registrazione degli accessi, tu, in qualità di API sviluppatore, desideri registrare chi ha effettuato l'accesso al tuo API e in che modo il chiamante ha effettuato l'accesso al. API Puoi creare il tuo gruppo di log o scegliere un gruppo di log esistente che possa essere gestito da API Gateway. Per specificare i dettagli di accesso, si selezionano `$context`le variabili, un formato di registro e una destinazione del gruppo di log.

Il formato del log di accesso deve includere almeno `$context.requestId` o `$context.extendedRequestId`. Come procedura consigliata, includi `$context.requestId` e `$context.extendedRequestId` nel formato di registro.

`$context.requestId`

Questo registra il valore nell'`x-amzn-RequestId`intestazione. I client possono sovrascrivere il valore nell'`x-amzn-RequestId`intestazione con un valore nel formato di un identificatore univoco universale (). UUID APIGateway restituisce questo ID di richiesta nell'intestazione della risposta. `x-amzn-RequestId` APIGateway sostituisce le richieste sovrascritte IDs che non sono in formato a *`UUID`*`_REPLACED_INVALID_REQUEST_ID` nei log UUID di accesso.

`$context.extendedRequestId`

L'extendedRequestID è un ID univoco generato da Gateway. API APIGateway restituisce questo ID di richiesta nell'intestazione della `x-amz-apigw-id` risposta. Un API chiamante non può fornire o sostituire questo ID di richiesta. Potrebbe essere necessario fornire questo valore a AWS Support per risolvere il problema. API Per ulteriori informazioni, consulta the section called "`$context`Variabili per modelli di dati, autorizzatori, modelli di mappatura e registrazione degli accessi CloudWatch ".

> **Note**
>
> Sono `$context` supportate solo le variabili.

Scegliete un formato di registro adottato anche dal backend analitico, ad esempio Common Log Format (CLF), JSONXML, o. CSV Quindi puoi alimentarlo direttamente con i log di accesso per l'elaborazione e il rendering dei parametri. Per definire il formato dei log, impostate il gruppo di log ARN sulla destinationArn proprietà accessLogSettings/sullo stage. È possibile ottenere un gruppo di log ARN nella CloudWatch console. Per definire il formato del registro degli accessi, impostate un formato scelto nella proprietà accessLogSetting/format sullo stage.

Esempi di alcuni formati di registro degli accessi di uso comune sono mostrati nella console API Gateway e sono elencati di seguito.

- `CLF` (Common Log Format):

```
$context.identity.sourceIp $context.identity.caller $context.identity.user
 [$context.requestTime]"$context.httpMethod $context.resourcePath
 $context.protocol" $context.status $context.responseLength $context.requestId
 $context.extendedRequestId
```

- `JSON`:

```
{ "requestId":"$context.requestId",
 "extendedRequestId":"$context.extendedRequestId","ip": "$context.identity.sourceIp",
 "caller":"$context.identity.caller", "user":"$context.identity.user",
 "requestTime":"$context.requestTime", "httpMethod":"$context.httpMethod",
 "resourcePath":"$context.resourcePath", "status":"$context.status",
 "protocol":"$context.protocol", "responseLength":"$context.responseLength" }
```

- `XML`:

```
<request id="$context.requestId"> <extendedRequestId>$context.extendedRequestId</
extendedRequestId> <ip>$context.identity.sourceIp</ip> <caller>
$context.identity.caller</caller> <user>$context.identity.user</user> <requestTime>
$context.requestTime</requestTime> <httpMethod>$context.httpMethod</httpMethod>
 <resourcePath>$context.resourcePath</resourcePath> <status>$context.status</status>
 <protocol>$context.protocol</protocol> <responseLength>$context.responseLength</
responseLength> </request>
```

- `CSV` (valori separati da virgola):

```
$context.identity.sourceIp,$context.identity.caller,$context.identity.user,
$context.requestTime,$context.httpMethod,$context.resourcePath,$context.protocol,
$context.status,$context.responseLength,$context.requestId,$context.extendedRequestId
```

Autorizzazioni per la registrazione CloudWatch

Per abilitare CloudWatch i log, devi concedere a API Gateway l'autorizzazione a leggere e scrivere i log del tuo account. CloudWatch La policy `AmazonAPIGatewayPushToCloudWatchLogs` gestita (con un ARN of`arn:aws:iam::aws:policy/service-role/AmazonAPIGatewayPushToCloudWatchLogs`) dispone di tutte le autorizzazioni richieste:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
```

```
                "logs:PutLogEvents",
                "logs:GetLogEvents",
                "logs:FilterLogEvents"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> APIII gateway chiama AWS Security Token Service per assumere il IAM ruolo, quindi assicurati che AWS STS sia abilitato per la regione. Per ulteriori informazioni, consulta Gestire AWS STS in una AWS regione.

Per concedere queste autorizzazioni al tuo account, crea un IAM ruolo `apigateway.amazonaws.com` come entità attendibile, allega la politica precedente al IAM ruolo e imposta il IAM ruolo nella proprietà ARN cloudWatchRoleArn del tuo account. È necessario impostare la proprietà cloudWatchRoleArn separatamente per ogni AWS regione in cui si desidera abilitare i registri. CloudWatch

Se ricevi un errore durante l'impostazione del IAM ruoloARN, controlla le impostazioni del tuo AWS Security Token Service account per assicurarti che AWS STS sia abilitato nella regione che stai utilizzando. Per ulteriori informazioni sull'abilitazione AWS STS, consulta Managing AWS STS in an AWS Region nella Guida IAM per l'utente.

Configurare la CloudWatch API registrazione utilizzando la console API Gateway

Per configurare la CloudWatch API registrazione, è necessario averla distribuita su una faseAPI. È inoltre necessario aver configurato un ruolo CloudWatch Logs appropriato ARN per il proprio account.

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel pannello di navigazione principale, scegli Impostazioni, quindi in Registrazione seleziona Modifica.
3. Per il ruolo di CloudWatch registro ARN, inserisci un IAM ruolo con le ARN autorizzazioni appropriate. È necessario eseguire questa operazione una volta per ogni Account AWS creazione creata APIs utilizzando API Gateway.
4. Nel riquadro di navigazione principale, scegli APIs, quindi esegui una delle seguenti operazioni:

a. Scegliete una fase esistenteAPI, quindi scegliete una fase.

b. Createne uno API e poi distribuitelo su uno stage.

5. Nel riquadro di navigazione principale scegli Fasi.

6. Nella sezione Log e tracciamento scegli Modifica.

7. Per abilitare il logging dell'esecuzione:

   a. Seleziona un livello di registrazione dal menu a discesa CloudWatch Registri. I livelli di registrazione sono i seguenti:

      - Disattivata: la registrazione non è attivata per questa fase.
      - Solo errori: la registrazione è abilitata solo per gli errori.
      - Errori e registri delle informazioni: la registrazione è abilitata per tutti gli eventi.

   b. (Facoltativo) Seleziona Tracciamento dati per attivare la registrazione della traccia dei dati per la tua fase. Questa operazione può essere utile per la risoluzione dei problemiAPIs, ma può comportare la registrazione di dati sensibili.

      > **Note**
      >
      > Ti consigliamo di non utilizzare il tracciamento dei dati per la produzione. APIs

   c. (Facoltativo) Seleziona Metriche dettagliate per attivare le metriche dettagliate CloudWatch .

   Per ulteriori informazioni sulle CloudWatch metriche, consulta. the section called “CloudWatch metriche”

8. Per abilitare il logging degli accessi:

   a. Attiva Registrazione accesso personalizzato.

   b. Per Destinazione del log di Access ARN, inserisci il nome ARN di un gruppo di log. Il ARN formato è`arn:aws:logs:{region}:{account-id}:log-group:log-group-name`.

   c. In Formato dei log immetti un formato di log. Puoi scegliere CLF, JSONXML, o CSV. Per ulteriori informazioni sui formati di log di esempio, consulta the section called “CloudWatch formati di registro per API Gateway”.

9. Scegli Save changes (Salva modifiche).

> **Note**
>
> Puoi abilitare la registrazione dell'esecuzione e quella degli accessi in modo reciprocamente indipendente.

APIGateway è ora pronto per registrare le richieste al tuoAPI. Non è necessario ridistribuire il API quando si aggiornano le impostazioni dello stage, i log o le variabili dello stage.

Configura la registrazione utilizzando CloudWatch API AWS CloudFormation

Utilizza il seguente AWS CloudFormation modello di esempio per creare un gruppo di log Amazon CloudWatch Logs e configurare l'esecuzione e la registrazione degli accessi per una fase. Per abilitare CloudWatch i log, devi concedere a API Gateway l'autorizzazione a leggere e scrivere i log del tuo account. CloudWatch Per ulteriori informazioni, consulta Associare l'account al IAM ruolo nella Guida per l'AWS CloudFormation utente.

```
  TestStage:
    Type: AWS::ApiGateway::Stage
    Properties:
      StageName: test
      RestApiId: !Ref MyAPI
      DeploymentId: !Ref Deployment
      Description: "test stage description"
      MethodSettings:
        - ResourcePath: "/*"
          HttpMethod: "*"
          LoggingLevel: INFO
      AccessLogSetting:
        DestinationArn: !GetAtt MyLogGroup.Arn
        Format: $context.extendedRequestId $context.identity.sourceIp
$context.identity.caller $context.identity.user [$context.requestTime]
"$context.httpMethod $context.resourcePath $context.protocol" $context.status
$context.responseLength $context.requestId
  MyLogGroup:
    Type: AWS::Logs::LogGroup
    Properties:
      LogGroupName: !Join
        - '-'
        - - !Ref MyAPI
          - access-logs
```

Registra le chiamate API REST su Amazon Data Firehose in API Gateway

Per risolvere i problemi relativi all'accesso dei client alla tua API, puoi registrare le chiamate API su Amazon Data Firehose. Per ulteriori informazioni su Firehose, consulta What Is Amazon Data Firehose? .

Per la registrazione degli accessi, puoi solo abilitare CloudWatch o FireHose, non puoi abilitare entrambi. Tuttavia, è possibile abilitare CloudWatch la registrazione dell'esecuzione e Firehose la registrazione degli accessi.

Argomenti

- Formati di registro Firehose per API Gateway
- Autorizzazioni per la registrazione di Firehose
- Configurare la registrazione degli accessi Firehose utilizzando la console API Gateway

Formati di registro Firehose per API Gateway

La registrazione di Firehose utilizza lo stesso formato della registrazione. CloudWatch

Autorizzazioni per la registrazione di Firehose

Quando la registrazione degli accessi Firehose è abilitata su uno stage, API Gateway crea un ruolo collegato al servizio nell'account, se il ruolo non esiste già. Il ruolo è denominato `AWSServiceRoleForAPIGateway` ed è collegato alla policy gestita `APIGatewayServiceRolePolicy`. Per ulteriori informazioni sui ruoli collegati ai servizi, consulta Utilizzo dei ruoli collegati ai servizi.

> **Note**
>
> Il nome dello stream Firehose deve essere. `amazon-apigateway-{your-stream-name}`

Configurare la registrazione degli accessi Firehose utilizzando la console API Gateway

Per configurare la registrazione API, devi aver distribuito l'API in una fase. È inoltre necessario aver creato uno stream Firehose.

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Esegui una di queste operazioni:
   a. Seleziona un'API esistente e quindi scegli una fase.
   b. Crea un'API e distribuiscila in una fase.
3. Nel riquadro di navigazione principale scegli Fasi.
4. Nella sezione Log e tracciamento scegli Modifica.
5. Per abilitare la registrazione degli accessi a uno stream Firehose:
   a. Attiva Registrazione accesso personalizzato.
   b. Per l'ARN di destinazione del registro di accesso, immettete l'ARN di uno stream Firehose. Il formato dell'ARN è `arn:aws:firehose:{region}:{account-id}:deliverystream/amazon-apigateway-{your-stream-name}`.

      > **Note**
      > Il nome dello stream Firehose deve essere. `amazon-apigateway-{your-stream-name}`

   c. In Formato dei log immetti un formato di log. Puoi scegliere CLF, JSON, XML o CSV. Per ulteriori informazioni sui formati di log di esempio, consulta the section called “CloudWatch formati di registro per API Gateway”.
6. Seleziona Salvataggio delle modifiche.

API Gateway è ora pronto per registrare le richieste nella tua API su Firehose. Non è necessario ridistribuire l'API quando si aggiornano le impostazioni delle fasi, i log o le variabili delle fasi.

Traccia le richieste degli utenti sulle API REST utilizzando X-Ray in API Gateway

Puoi utilizzare AWS X-Ray per monitorare e analizzare le richieste degli utenti durante il passaggio attraverso le API REST di Amazon API Gateway verso i servizi sottostanti. API Gateway supporta il monitoraggio attraverso X-Ray per tutti i tipi di endpoint API REST di API Gateway: regionale, ottimizzato per l’edge e privato. Puoi usare X-Ray con Amazon API Gateway in tutte le AWS regioni in cui X-Ray è disponibile.

Poiché X-Ray offre una end-to-end visualizzazione dell'intera richiesta, è possibile analizzare le latenze nelle API e nei relativi servizi di backend. È possibile utilizzare una mappa del servizio di X-Ray per visualizzare la latenza di un'intera richiesta e quella dei servizi downstream integrati con X-Ray. Inoltre, puoi configurare le regole di campionamento per comunicare a X-Ray quali richieste registrare e a quale frequenza in base alle policy specificate.

Se chiami un'API di API Gateway da un servizio già tracciato, API Gateway passa subito attraverso il monitoraggio, anche se X-Ray non è abilitato sull'API.

È possibile abilitare X-Ray per una fase API utilizzando la console API Gateway o utilizzando l'API o l'interfaccia a riga di comando di API Gateway.

Argomenti

- Configurazione AWS X-Ray con le API REST di API Gateway
- Usa le mappe dei AWS X-Ray servizi e traccia le viste con API Gateway
- Configurazione delle regole AWS X-Ray di campionamento per le API API Gateway
- AWS X-Ray trace per le API di Amazon API Gateway

Configurazione AWS X-Ray con le API REST di API Gateway

In questa sezione puoi trovare informazioni dettagliate su come configurare AWS X-Ray con le API REST di API Gateway.

Argomenti

- Modalità di monitoraggio di X-Ray per API Gateway
- Autorizzazioni per il monitoraggio di X-Ray
- Abilitazione del monitoraggio tramite X-Ray nella console API Gateway
- Abilitazione del AWS X-Ray tracciamento utilizzando l'API Gateway CLI

Modalità di monitoraggio di X-Ray per API Gateway

Il percorso di una richiesta nell'applicazione viene tracciato mediante un ID traccia. Una traccia raccoglie tutti i segmenti generati da una singola richiesta, in genere una richiesta HTTP `GET` o `POST`.

Per un'API di API Gateway sono disponibili due modalità di monitoraggio:

- Passiva: questa è l'impostazione predefinita se non hai abilitato il monitoraggio X-Ray su una fase API. Questo approccio significa che l'API di API Gateway viene tracciata solo se X-Ray è stato abilitato in un servizio upstream.
- Attiva: quando per una fase API di API Gateway si sceglie questa impostazione, API Gateway esegue automaticamente il campionamento delle richieste di invocazione dell'API in base all'algoritmo di campionamento specificato da X-Ray.

  Quando su una fase il monitoraggio è abilitato in modalità attiva, API Gateway crea un ruolo collegato ai servizi nel tuo account se questo non è già esistente. Il ruolo è denominato `AWSServiceRoleForAPIGateway` e sarà collegato alla policy gestita `APIGatewayServiceRolePolicy`. Per ulteriori informazioni sui ruoli collegati ai servizi, consulta Utilizzo dei ruoli collegati ai servizi.

  > **Note**
  >
  > X-Ray applica un algoritmo di campionamento per garantire che il monitoraggio avvenga in modo efficiente, continuando allo stesso tempo a fornire un campione rappresentativo delle richieste ricevute dall'API. L'algoritmo di campionamento di default corrisponde a una richiesta al secondo, con il 5% di richieste campionate oltre tale limite.

Puoi modificare la modalità di tracciamento per la tua API utilizzando la console di gestione API Gateway, l'API Gateway CLI o AWS un SDK.

Autorizzazioni per il monitoraggio di X-Ray

Quando su una fase è abilitato il monitoraggio tramite X-Ray, API Gateway crea un ruolo collegato ai servizi nel tuo account se questo non è già esistente. Il ruolo è denominato `AWSServiceRoleForAPIGateway` e sarà collegato alla policy gestita `APIGatewayServiceRolePolicy`. Per ulteriori informazioni sui ruoli collegati ai servizi, consulta Utilizzo dei ruoli collegati ai servizi.

Abilitazione del monitoraggio tramite X-Ray nella console API Gateway

Puoi utilizzare la console di Amazon API Gateway per abilitare il monitoraggio in modalità attiva in una fase API.

In queste fasi si presuppone che l'API sia già stata distribuita a una fase.

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.

2. Scegli l'API, quindi nel pannello di navigazione principale, seleziona Fasi.
3. Nel riquadro Fasi scegli una fase.
4. Nella sezione Log e tracciamento scegli Modifica.
5. Per abilitare il tracciamento X-Ray seleziona Tracciamento X-Ray per attivarlo.
6. Seleziona Salvataggio delle modifiche.

Una volta abilitato X-Ray per la fase API, è possibile utilizzare la console di gestione di X-Ray per visualizzare il monitoraggio e le mappe di servizio.

Abilitazione del AWS X-Ray tracciamento utilizzando l'API Gateway CLI

Per utilizzare l'opzione AWS CLI per abilitare il tracciamento X-Ray attivo per una fase API quando create la fase, chiamate il comando create-stage come nell'esempio seguente:

```
aws apigateway create-stage \
    --rest-api-id {rest-api-id} \
    --stage-name {stage-name} \
    --deployment-id {deployment-id} \
    --region {region} \
    --tracing-enabled=true
```

Di seguito è riportato un esempio di output per una chiamata che va a buon fine:

```
{
    "tracingEnabled": true,
    "stageName": {stage-name},
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": {deployment-id},
    "lastUpdatedDate": 1533849811,
    "createdDate": 1533849811,
    "methodSettings": {}
}
```

Per utilizzare AWS CLI per disabilitare il tracciamento X-Ray attivo per una fase API quando create la fase, chiamate il comando create-stage come nell'esempio seguente:

```
aws apigateway create-stage \
```

```
    --rest-api-id {rest-api-id} \
    --stage-name {stage-name} \
    --deployment-id {deployment-id} \
    --region {region} \
    --tracing-enabled=false
```

Di seguito è riportato un esempio di output per una chiamata che va a buon fine:

```
{
    "tracingEnabled": false,
    "stageName": {stage-name},
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": {deployment-id},
    "lastUpdatedDate": 1533849811,
    "createdDate": 1533849811,
    "methodSettings": {}
}
```

Per utilizzare AWS CLI per abilitare il tracciamento X-Ray attivo per un'API già distribuita, chiamate il comando update-stage come segue:

```
aws apigateway update-stage \
    --rest-api-id {rest-api-id} \
    --stage-name {stage-name} \
    --patch-operations op=replace,path=/tracingEnabled,value=true
```

Per utilizzare per AWS CLI disabilitare il tracciamento X-Ray attivo per un'API che è già stata distribuita, chiamate il comando update-stage come nell'esempio seguente:

```
aws apigateway update-stage \
    --rest-api-id {rest-api-id} \
    --stage-name {stage-name} \
    --region {region} \
    --patch-operations op=replace,path=/tracingEnabled,value=false
```

Di seguito è riportato un esempio di output per una chiamata che va a buon fine:

```
{
    "tracingEnabled": false,
```

```
    "stageName": {stage-name},
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": {deployment-id},
    "lastUpdatedDate": 1533850033,
    "createdDate": 1533849811,
    "methodSettings": {}
}
```

Una volta abilitato X-Ray per la fase API, utilizzare l'interfaccia a riga di comando di X-Ray per recuperare le informazioni sul monitoraggio. Per ulteriori informazioni, vedere Utilizzo dell'API X-Ray con la CLI AWS.

Usa le mappe dei AWS X-Ray servizi e traccia le viste con API Gateway

In questa sezione sono disponibili informazioni dettagliate sull'utilizzo delle mappe di servizio e delle viste di monitoraggio di AWS X-Ray con API Gateway.

Argomenti

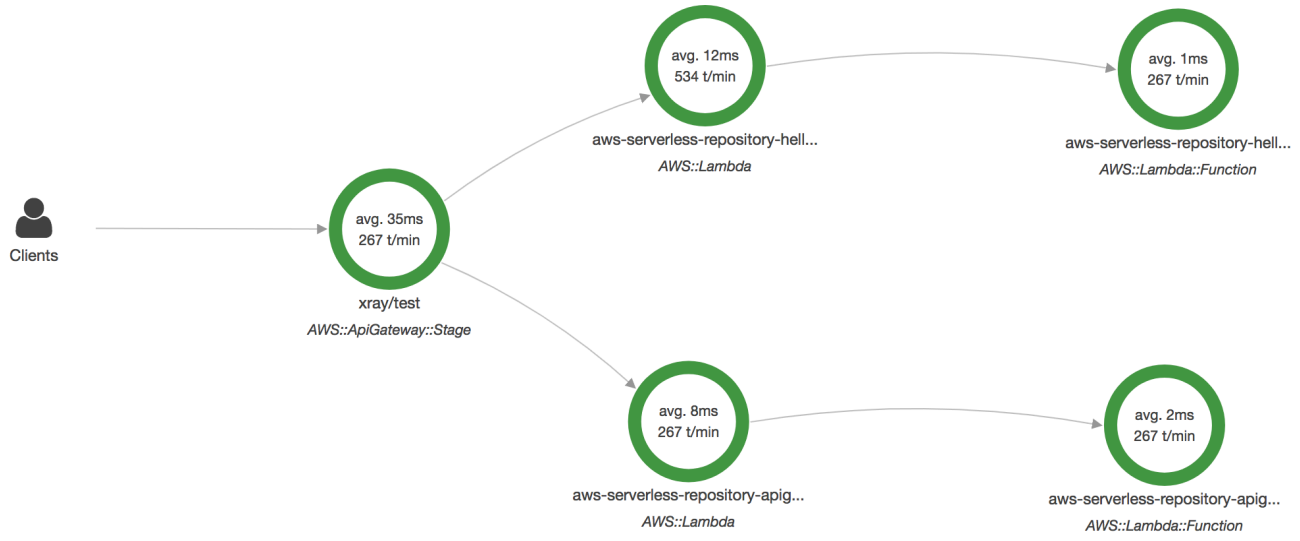
- Esempio di mappa dei servizi di X-Ray
- Esempio di visualizzazione del monitoraggio di X-Ray

Esempio di mappa dei servizi di X-Ray

AWS X-Ray le mappe dei servizi mostrano informazioni sull'API e su tutti i relativi servizi a valle. Quando X-Ray è abilitato per una fase API in API Gateway, sarà possibile visualizzare un nodo nella mappa di servizio che contiene informazioni sul tempo complessivo passato nel servizio API Gateway. Puoi ottenere informazioni dettagliate sullo stato della risposta e un istogramma del tempo di risposta dell'API nel periodo di tempo selezionato. Per le API che si integrano con AWS servizi come AWS Lambda Amazon DynamoDB, vedrai più nodi che forniscono metriche prestazionali relative a tali servizi. C'è una mappa di servizio per ogni fase API.

L'esempio seguente mostra una mappa di servizio per la fase `test` di un'API denominata `xray`. Questa API dispone di un'integrazione Lambda con una funzione di autorizzazione Lambda e una funzione di back-end Lambda. I nodi rappresentano il servizio API Gateway, il servizio Lambda e le due funzioni Lambda.

Per una spiegazione dettagliata della struttura della mappa dei servizi, vedere Utilizzare la mappa di tracciamento a raggi X..



La mappa dei servizi può essere ingrandita per mostrare una vista della traccia della fase API. La traccia visualizza informazioni approfondite riguardanti l'API, rappresentate come segmenti e segmenti secondari. Ad esempio, il monitoraggio della mappa di servizio mostrata sopra include i segmenti del servizio Lambda e della funzione Lambda. Per ulteriori informazioni, vedere AWS Lambda e AWS X-Ray.

Se scegli un nodo o un edge su una mappa di servizio di X-Ray, la console di X-Ray mostra un istogramma della distribuzione della latenza. Puoi utilizzare un istogramma della latenza per visualizzare il tempo impiegato da un servizio per completare le richieste. Di seguito è riportato un istogramma della fase dell'API Gateway denominata `xray/test` nella precedente mappa di servizio. Per una spiegazione dettagliata degli istogrammi di distribuzione della latenza, consulta Utilizzare gli istogrammi di latenza.

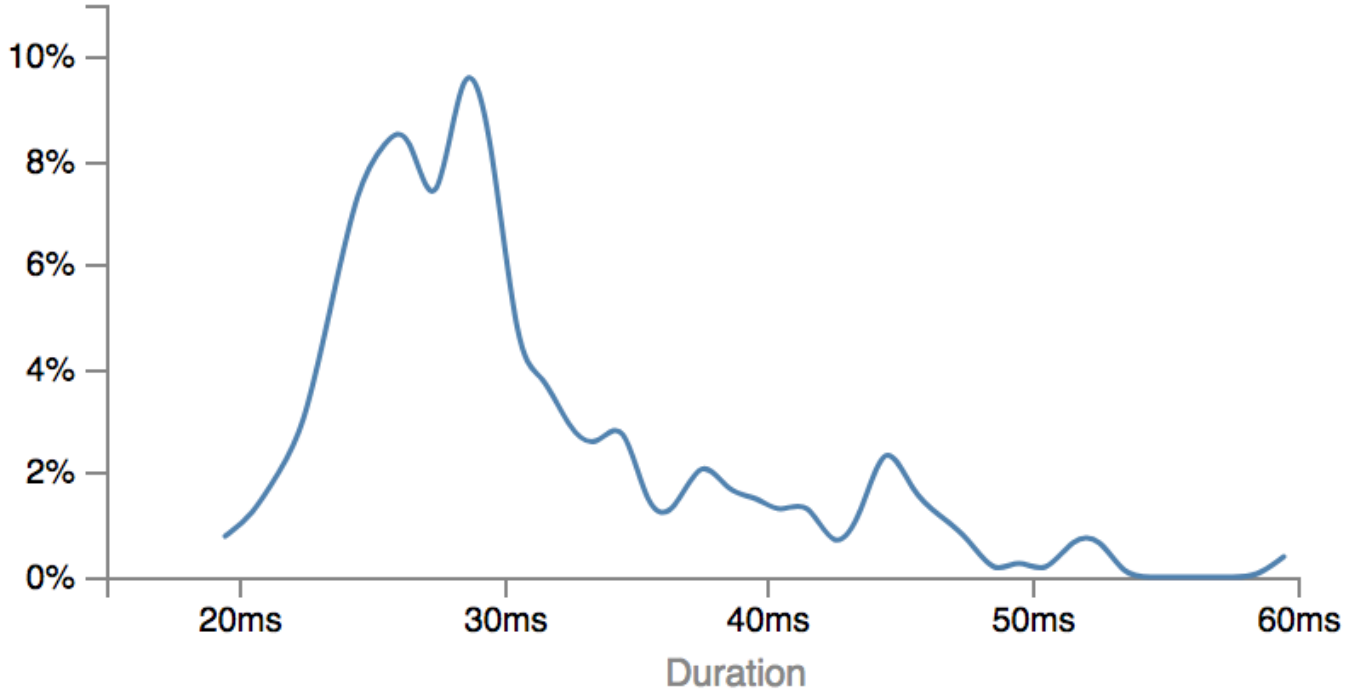
Service details

Name: xray/test

Type: AWS::ApiGateway::Stage

Response distribution

Click and drag to select an area to zoom in on or use as a latency filter when viewing traces.



Response status

Choose response statuses to add to the filter when viewing traces.

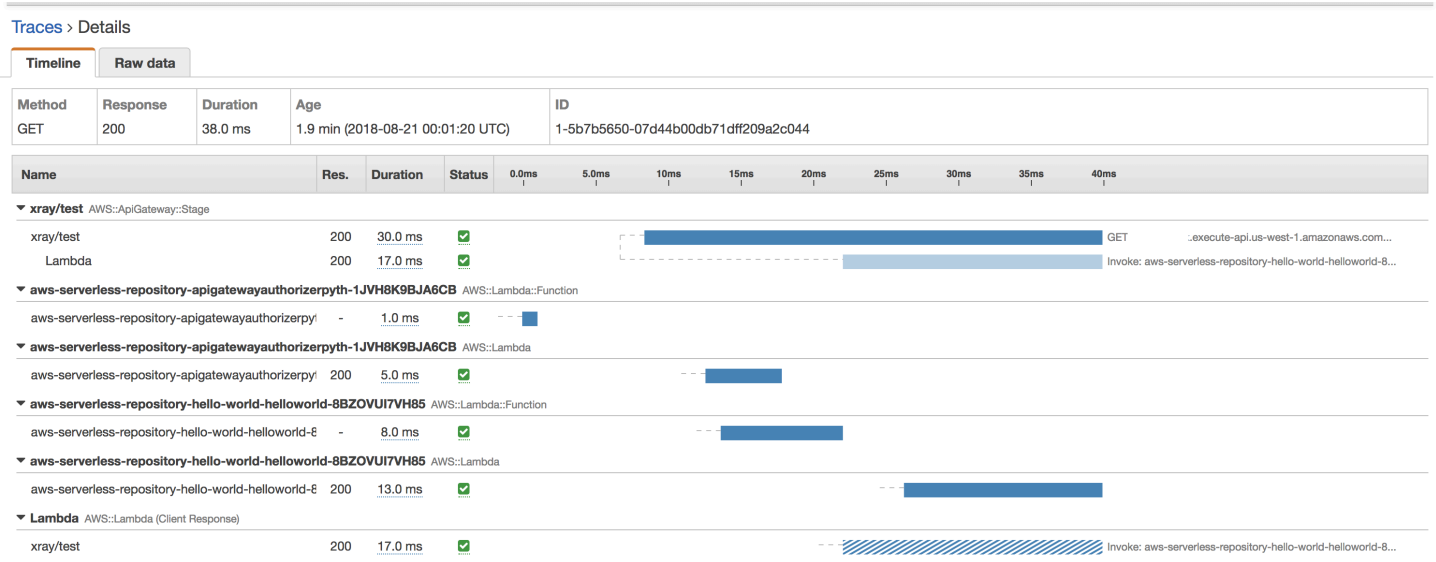
OK: 100% Error: 0%

Fault: 0% Throttle: 0%

Esempio di visualizzazione del monitoraggio di X-Ray

Il seguente diagramma mostra una vista del monitoraggio generato per l'API di esempio descritta in precedenza, con una funzione di back-end Lambda e una funzione di autorizzazione Lambda. Viene mostrata una richiesta metodo API andata a buon fine con codice di risposta pari a 200.

Per una spiegazione dettagliata delle viste di traccia, consulta Visualizzare le tracce e i dettagli di traccia.



Configurazione delle regole AWS X-Ray di campionamento per le API API Gateway

Puoi utilizzare la AWS X-Ray console o l'SDK per configurare le regole di campionamento per la tua API Amazon API Gateway. Una regola di campionamento specifica le richieste che X-Ray deve registrare per l'API. Personalizzando le regole di campionamento è possibile controllare la quantità di dati da registrare e modificare immediatamente il campionamento senza dover cambiare o ridistribuire il codice.

Prima di specificare le regole di campionamento di X-Ray, leggere i seguenti argomenti nella Guida per gli sviluppatori di X-Ray:

- Configura le regole di campionamento
- Utilizzo delle regole di campionamento con l'API X-Ray

Argomenti

- Valori delle opzioni delle regole di campionamento di X-Ray per le API di API Gateway
- Esempi di regole di campionamento di X-Ray

Valori delle opzioni delle regole di campionamento di X-Ray per le API di API Gateway

Le seguenti opzioni di campionamento di X-Ray sono rilevanti per API Gateway. I valori di stringa possono utilizzare caratteri jolly per corrispondere a un solo carattere (?) o a zero o più caratteri (*). Per ulteriori dettagli, inclusa una spiegazione dettagliata di come vengono utilizzate le impostazioni Reservoir e Rate, consulta Configura le regole di campionamento.

- Nome regola (stringa): un nome univoco per la regola.
- Priorità (numero intero compreso tra 1 e 9999): la priorità della regola di campionamento. I servizi valutano le regole in ordine crescente di priorità e prendono una decisione sul campionamento in base alla prima regola corrispondente.
- Riserva (numero intero non negativo): un numero fisso di richieste che rispettano il filtro da analizzare ogni secondo, prima di applicare la percentuale fissa. Il reservoir non viene utilizzato direttamente dai servizi, ma si applica a tutti i servizi che utilizzano la regola nel loro complesso.
- Percentuale (numero tra 0 e 100): la percentuale di richieste che rispettano il filtro da analizzare dopo l'esaurimento della riserva.
- Nome servizio (stringa): nome della fase API, nel formato ***{api-name}/{stage-name}***. Ad esempio, se dovessi distribuire l'API di PetStoreesempio in una fase denominata`test`, il valore del nome del servizio da specificare nella regola di campionamento sarebbe. **`pets/test`**
- Tipo servizio (stringa): per un'API di API Gateway è possibile specificare sia **`AWS::ApiGateway::Stage`** che **`AWS::ApiGateway::*`**.
- Host (stringa): il nome host ricavato dall'intestazione HTTP host. Per la corrispondenza a tutti i nome host, va impostato su **`*`**. Si può anche specificare un nome host completo o parziale di corrispondenza, ad esempio **`api.example.com`** o **`*.example.com`**.
- Resource ARN (ARN risorsa) (stringa): l'ARN della fase API, ad esempio **`arn:aws:apigateway:`*`region`*`::/restapis/`*`api-id`*`/stages/`*`stage-name`***.

  Il nome di fase si può ricavare dalla console, dall'interfaccia a riga di comando o dall'API di API Gateway. Per ulteriori informazioni sui formati degli ARN, consulta Riferimenti generali di Amazon Web Services.
- Metodo HTTP (stringa): il metodo da campionare, ad esempio **`GET`**.
- URL path (Percorso URL) (stringa): il percorso dell'URL della richiesta.

- (opzionale) Attributi (chiave e valore): intestazioni provenienti dalla richiesta HTTP originale, ad esempio **`Connection`**, **`Content-Length`** o **`Content-Type`**. Ogni valore dell'attributo può contenere fino a 32 caratteri.

Esempi di regole di campionamento di X-Ray

Esempio 1 di regola di campionamento

Questa regola campiona tutte le richieste `GET` per l'API `testxray` nella fase `test`.

- Nome regola — **`test-sampling`**
- Priorità — **`17`**
- Dimensioni riserva — **`10`**
- Percentuale fissa — **`10`**
- Nome servizio — **`testxray/test`**
- Tipo servizio — **`AWS::ApiGateway::Stage`**
- Metodo HTTP — **`GET`**
- ARN risorsa — **`*`**
- Host — **`*`**

Esempio 2 di regola di campionamento

Questa regola campiona tutte le richieste `testxray` per l'API nella fase `prod`.

- Nome regola — **`prod-sampling`**
- Priorità — **`478`**
- Dimensioni riserva — **`1`**
- Percentuale fissa — **`60`**
- Nome servizio — **`testxray/prod`**
- Tipo servizio — **`AWS::ApiGateway::Stage`**
- Metodo HTTP — **`*`**
- ARN risorsa — **`*`**
- Host — **`*`**
- Attributi — **`{}`**

AWS X-Ray trace per le API di Amazon API Gateway

Questa sezione illustra i segmenti di AWS X-Ray traccia, i sottosegmenti e altri campi di traccia per le API di Amazon API Gateway.

Prima di leggere questa sezione, rivedere i seguenti argomenti nella Guida per gli sviluppatori di X-Ray:

- Usa un AWS Management Console
- Documenti relativi al segmento X-Ray
- Concetti

Argomenti

- Esempi di oggetti di monitoraggio per un'API di API Gateway
- Comprendere la traccia

Esempi di oggetti di monitoraggio per un'API di API Gateway

Questa sezione illustra alcuni degli oggetti visibili in un monitoraggio di un'API di API Gateway.

Annotazioni

Le annotazioni possono essere visualizzate nei segmenti e nei segmenti secondari. Servono da espressioni di filtro nelle regole di campionamento per filtrare le tracce. Per ulteriori informazioni, consulta Configurare le regole di campionamento.

Di seguito è riportato un esempio di un oggetto `annotations`, in cui una fase API è identificata dall'ID dell'API e dal nome della fase API:

```
"annotations": {
    "aws:api_id": "a1b2c3d4e5",
    "aws:api_stage": "dev"
}
```

Per ulteriori informazioni sulle annotazioni, consultate Documenti con segmenti a raggi X, quindi scegliete Documenti con segmenti a raggi X, Annotazioni.

AWS dati relativi alle risorse

L'oggetto `aws` viene visualizzato solo nei segmenti. Di seguito è riportato l'esempio di un oggetto `aws` corrispondente alla regola di campionamento predefinita. Per una spiegazione approfondita delle regole di campionamento, consulta Configurare le regole di campionamento.

```
"aws": {
    "xray": {
        "sampling_rule_name": "Default"
    },
    "api_gateway": {
        "account_id": "123412341234",
        "rest_api_id": "a1b2c3d4e5",
        "stage": "dev",
        "request_id": "a1b2c3d4-a1b2-a1b2-a1b2-a1b2c3d4e5f6"
    }
}
```

Per ulteriori informazioni sull'**`aws`**oggetto, consultate Documenti dei segmenti a raggi X, quindi scegliete Documenti dei segmenti a raggi X, dati delle risorse.AWS

Comprendere la traccia

Di seguito è riportato un segmento di monitoraggio per una fase dell'API Gateway. Per una spiegazione dettagliata dei campi che compongono il segmento di traccia, consultate Documenti sui segmenti X-Ray.

```
        {
            "Document": {
                "id": "a1b2c3d4a1b2c3d4",
                "name": "testxray/dev",
                "start_time": 1533928226.229,
                "end_time": 1533928226.614,
                "metadata": {
                    "default": {
                        "extended_request_id": "abcde12345abcde=",
                        "request_id": "a1b2c3d4-a1b2-a1b2-a1b2-a1b2c3d4e5f6"
                    }
                },
                "http": {
                    "request": {
                        "url": "https://example.com/dev?
username=demo&message=hellofromdemo/",
                        "method": "GET",
```

```
                        "client_ip": "192.0.2.0",
                        "x_forwarded_for": true
                    },
                    "response": {
                        "status": 200,
                        "content_length": 0
                    }
                },
                "aws": {
                    "xray": {
                        "sampling_rule_name": "Default"
                    },
                    "api_gateway": {
                        "account_id": "123412341234",
                        "rest_api_id": "a1b2c3d4e5",
                        "stage": "dev",
                        "request_id": "a1b2c3d4-a1b2-a1b2-a1b2-a1b2c3d4e5f6"
                    }
                },
                "annotations": {
                    "aws:api_id": "a1b2c3d4e5",
                    "aws:api_stage": "dev"
                },
                "trace_id": "1-a1b2c3d4-a1b2c3d4a1b2c3d4a1b2c3d4",
                "origin": "AWS::ApiGateway::Stage",
                "resource_arn": "arn:aws:apigateway:us-east-1::/restapis/a1b2c3d4e5/
stages/dev",
                "subsegments": [
                    {
                        "id": "abcdefgh12345678",
                        "name": "Lambda",
                        "start_time": 1533928226.233,
                        "end_time": 1533928226.6130002,
                        "http": {
                            "request": {
                                "url": "https://example.com/2015-03-31/functions/
arn:aws:lambda:us-east-1:123412341234:function:xray123/invocations",
                                "method": "GET"
                            },
                            "response": {
                                "status": 200,
                                "content_length": 62
                            }
                        },
```

```
                        "aws": {
                            "function_name": "xray123",
                            "region": "us-east-1",
                            "operation": "Invoke",
                            "resource_names": [
                                "xray123"
                            ]
                        },
                        "namespace": "aws"
                    }
                ]
            },
            "Id": "a1b2c3d4a1b2c3d4"
        }
```

API HTTP API Gateway

REST API e API HTTP sono entrambe prodotti dell'API RESTful. Le REST API supportano più funzionalità rispetto alle API HTTP, mentre le API HTTP sono progettate con funzionalità minime in modo che possano essere offerte a un prezzo inferiore. Per ulteriori informazioni, consulta the section called “Scegli tra REST APIs e HTTP APIs ”.

È possibile utilizzare le API HTTP per inviare richieste a AWS Lambda funzioni o a qualsiasi endpoint HTTP instradabile. Ad esempio, puoi creare un'API HTTP che si integra con una funzione Lambda sul back-end. Quando un client chiama l'API, API Gateway invia la richiesta alla funzione Lambda e restituisce la risposta della funzione al client.

API HTTP supportano l'autorizzazione OpenID Connect e OAuth 2.0. Sono dotate di supporto integrato per CORS (cross-origin resource sharing) e le distribuzioni automatiche.

È possibile creare API HTTP utilizzando la Console di AWS gestione, le API o gli AWS CLI SDK. AWS CloudFormation

Argomenti

- Sviluppa API HTTP in API Gateway
- Pubblica API HTTP che i clienti possono richiamare
- Proteggi le tue API HTTP in API Gateway
- Monitora le API HTTP in API Gateway
- Risoluzione dei problemi relativi alle API HTTP in API Gateway

Sviluppa API HTTP in API Gateway

In questa sezione vengono fornite informazioni dettagliate sulle funzionalità di API Gateway necessarie durante lo sviluppo delle API di API Gateway.

Durante lo sviluppo delle API di API Gateway è possibile impostare una serie di caratteristiche dell'API. Queste caratteristiche dipendono dal caso d'uso dell'API. Ad esempio, è possibile consentire a solo a determinati client di richiamare l'API oppure che questa sia disponibile per tutti. È possibile decidere che una chiamata API esegua una funzione Lambda, una query a un database o richiami un'applicazione.

Argomenti

- Crea un'API HTTP
- Crea percorsi per le API HTTP in API Gateway
- Controlla e gestisci l'accesso alle API HTTP in API Gateway
- Crea integrazioni per le API HTTP in API Gateway
- Configurazione CORS per HTTP APIs in API Gateway
- Trasforma API le richieste e le risposte per HTTP APIs in API Gateway
- Usa le definizioni OpenAPI per le API HTTP in API Gateway

Crea un'API HTTP

Per creare un'API funzionale, è necessario disporre di almeno una route, integrazione, fase e distribuzione.

Gli esempi seguenti mostrano come creare un'API con un'integrazione AWS Lambda o HTTP, una route e una fase predefinita configurata per distribuire automaticamente le modifiche.

In questa guida si presuppone che tu abbia già familiarità con API Gateway e Lambda. Per una guida più dettagliata, consulta Inizia a usare.

Argomenti

- Crea un'API HTTP utilizzando il AWS Management Console
- Crea un'API HTTP utilizzando la AWS CLI

Crea un'API HTTP utilizzando il AWS Management Console

1. Apri la console API Gateway.
2. Seleziona Create API (Crea API).
3. In HTTP API, scegliere Build (Compila).
4. Scegliere Add integration (Aggiungi integrazione), quindi scegliere una funzione AWS Lambda o immettere un endpoint HTTP.
5. Per Name (Nome) immetti un nome per il TAG.
6. Scegliere Review and create (Rivedi e crea).

7. Seleziona Crea.

Ora la tua API è pronta per effettuare le chiamate. È possibile testare l'API inserendo il relativo URL di chiamata in un browser o utilizzando Curl.

```
curl https://api-id.execute-api.us-east-2.amazonaws.com
```

Crea un'API HTTP utilizzando la AWS CLI

È possibile utilizzare la creazione rapida per creare un'API con un'integrazione Lambda o HTTP, una route catch-all di default e una fase predefinita configurata per implementare automaticamente le modifiche. Il comando seguente utilizza la funzione di creazione rapida per creare un'API che si integra con una funzione Lambda sul back-end.

> **Note**
>
> Per richiamare un'integrazione Lambda, API Gateway deve disporre delle autorizzazioni necessarie. È possibile utilizzare una policy basata sulle risorse o un ruolo IAM per concedere le autorizzazioni di API Gateway per richiamare una funzione Lambda. Per ulteriori informazioni, consulta AWS Lambda Autorizzazioni nella Guida per gli AWS Lambda sviluppatori.

Example

```
aws apigatewayv2 create-api --name my-api --protocol-type HTTP --target
 arn:aws:lambda:us-east-2:123456789012:function:function-name
```

Ora la tua API è pronta per effettuare le chiamate. È possibile testare l'API inserendo il relativo URL di chiamata in un browser o utilizzando Curl.

```
curl https://api-id.execute-api.us-east-2.amazonaws.com
```

Crea percorsi per le API HTTP in API Gateway

Esegui il routing delle richieste API in entrata dirette alle risorse di back-end. Le route sono costituite da due parti: un metodo HTTP e un percorso delle risorse, ad esempi, `GET /pets`. È possibile definire metodi HTTP specifici per il percorso. In alternativa, è possibile utilizzare il metodo `ANY` per

abbinare tutti i metodi non definiti per una risorsa. È possibile creare una route `$default` che funga da catch-all per le richieste che non corrispondono ad altre route.

> **Note**
>
> Gateway Amazon API decodifica i parametri di richiesta con codifica URL prima di passarli all'integrazione back-end.

Lavorare con le variabili di percorso

Nelle route delle API HTTP è possibile utilizzare delle variabili di percorso.

Ad esempio, il percorso `GET /pets/{petID}` cattura una richiesta `GET` inviata da un client `https://api-id.execute-api.us-east-2.amazonaws.com/pets/6`.

Una variabile di percorso greedy cattura tutte le risorse figlio di un percorso. Per creare una variabile di percorso greedy, aggiungere + al nome della variale, ad esempio `{proxy+}`. La variabile di percorso greedy deve trovarsi alla fine del percorso della risorsa.

Utilizzo dei parametri della stringa di query

Per impostazione predefinita, API Gateway invia i parametri della stringa di query all'integrazione back-end se sono inclusi in una richiesta a un oggetto API HTTP.

Ad esempio, quando un client invia una richiesta a `https://api-id.execute-api.us-east-2.amazonaws.com/pets?id=4&type=dog`, i parametri della stringa di query `?id=4&type=dog` vengono inviati all'integrazione.

Lavorare con il percorso **`$default`**

Il percorso `$default` cattura le richieste che non corrispondono esplicitamente ad altri percorsi nell'API.

Quando la route `$default` riceve una richiesta, API Gateway invia il percorso completo della richiesta all'integrazione. Ad esempio, è possibile creare un'API con solo un percorso `$default` e integrarlo nel metodo `ANY` con l'endpoint `https://petstore-demo-endpoint.execute-api.com` HTTP. Quando si invia una richiesta a `https://api-id.execute-api.us-east-2.amazonaws.com/store/checkout`, API Gateway invia una richiesta a `https://petstore-demo-endpoint.execute-api.com/store/checkout`.

Per ulteriori informazioni sulle integrazioni HTTP, consultare Crea integrazioni proxy HTTP per API HTTP.

Routing delle richieste API

Quando un client invia una richiesta API, API Gateway stabilisce innanzitutto a quale fase instradare la richiesta. Se la richiesta corrisponde esplicitamente a una fase, API Gateway invia la richiesta a tale fase. Se nessuna fase corrisponde completamente alla richiesta, API Gateway invia la richiesta alla fase `$default`. Se non è presente alcuna `$default` fase, l'API restituisce `{"message":"Not Found"}` e non genera CloudWatch log.

Dopo aver selezionato una fase, API Gateway seleziona una route. API Gateway seleziona la route con la corrispondenza più specifica, utilizzando le seguenti priorità:

1. Corrispondenza completa per un percorso e un metodo.
2. Corrispondenza per un percorso e un metodo con una variabile di percorso greedy (`{proxy+}`).
3. Instradamento `$default`.

Se nessuna route corrisponde a una richiesta, API Gateway restituisce al client il valore `{"message":"Not Found"}`.

Ad esempio, si consideri un'API con una fase `$default` e le seguenti route di esempio:

1. `GET /pets/dog/1`
2. `GET /pets/dog/{id}`
3. `GET /pets/{proxy+}`
4. `ANY /{proxy+}`
5. `$default`

Nella tabella seguente viene riepilogato il modo in cui API Gateway instrada le richieste alle route di esempio.

Richiesta	Percorso selezionato	Spiegazione
`GET https://`*`api-id`*`.execute-`	`GET /pets/dog/1`	La richiesta corrisponde completamente a questo instradamento statico.

Richiesta	Percorso selezionato	Spiegazione
`api.` *`region`*`.amazonaws.com/pets/dog/1`		
`GET https://`*`api-id`*`.execute-api.` *`region`*`.amazonaws.com/pets/dog/2`	`GET /pets/dog/{id}`	La richiesta corrisponde completamente a questo percorso.
`GET https://`*`api-id`*`.execute-api.` *`region`*`.amazonaws.com/pets/cat/1`	`GET /pets/{proxy+}`	La richiesta non corrisponde completamente a un percorso. Il percorso con un metodo `GET` e una variabile di percorso greedy cattura questa richiesta .
`POST https://`*`api-id`*`.execute-api.` *`region`*`.amazonaws.com/test/5`	`ANY /{proxy+}`	Il metodo `ANY` corrisponde a tutti i metodi che non sono stati definiti per un percorso. I percorsi con variabili di percorso greedy hanno priorità più alta rispetto al percorso `$default`.

Controlla e gestisci l'accesso alle API HTTP in API Gateway

API Gateway supporta più meccanismi per controllare e gestire l'accesso a un'API HTTP:

- Le autorizzazioni Lambda utilizzano funzioni Lambda per controllare l'accesso alle API. Per ulteriori informazioni, consulta Controlla l'accesso HTTP APIs con gli AWS Lambda autorizzatori.
- Le autorizzazioni JWT utilizzano i token Web JSON per controllare l'accesso alle API. Per ulteriori informazioni, consulta Controlla l'accesso HTTP APIs con gli JWT autorizzatori in API Gateway.
- I ruoli e le policy AWS IAM standard offrono controlli di accesso flessibili e robusti. È possibile utilizzare ruoli e policy di IAM per controllare chi può creare e gestire le API e chi può invocarle. Per ulteriori informazioni, consulta Controlla l'accesso HTTP APIs con IAM autorizzazione in API Gateway.

Controlla l'accesso HTTP APIs con gli AWS Lambda autorizzatori

Utilizzi un autorizzatore Lambda per utilizzare una funzione Lambda per controllare l'accesso al tuo. HTTP API Quindi, quando un client chiama la tuaAPI, API Gateway richiama la tua funzione Lambda. APIGateway utilizza la risposta della tua funzione Lambda per determinare se il client può accedere alla tua. API

Tipo di formato payload

La versione del formato del payload dell'autorizzazione specifica il formato dei dati che API Gateway invia a un sistema di autorizzazione Lambda e il modo in cui Gateway API interpreta la risposta di Lambda. Se non si specifica una versione del formato di payload, utilizza la versione più recente per impostazione predefinita. AWS Management Console Se crei un autorizzatore Lambda utilizzando AWS CLI, o an AWS CloudFormation SDK, devi specificare un. `authorizerPayloadFormatVersion` I valori supportati sono `1.0` e `2.0`.

Se hai bisogno di compatibilità con RESTAPIs, usa la versione. `1.0`

Gli esempi seguenti mostrano la struttura di ogni tipo di formato payload.

2.0

```
{
  "version": "2.0",
  "type": "REQUEST",
  "routeArn": "arn:aws:execute-api:us-east-1:123456789012:abcdef123/test/GET/
request",
  "identitySource": ["user1", "123"],
  "routeKey": "$default",
  "rawPath": "/my/path",
  "rawQueryString": "parameter1=value1&parameter1=value2&parameter2=value",
  "cookies": ["cookie1", "cookie2"],
  "headers": {
    "header1": "value1",
    "header2": "value2"
  },
  "queryStringParameters": {
    "parameter1": "value1,value2",
    "parameter2": "value"
  },
  "requestContext": {
    "accountId": "123456789012",
```

```
    "apiId": "api-id",
    "authentication": {
      "clientCert": {
        "clientCertPem": "CERT_CONTENT",
        "subjectDN": "www.example.com",
        "issuerDN": "Example issuer",
        "serialNumber": "1",
        "validity": {
          "notBefore": "May 28 12:30:02 2019 GMT",
          "notAfter": "Aug  5 09:36:04 2021 GMT"
        }
      }
    },
    "domainName": "id.execute-api.us-east-1.amazonaws.com",
    "domainPrefix": "id",
    "http": {
      "method": "POST",
      "path": "/my/path",
      "protocol": "HTTP/1.1",
      "sourceIp": "IP",
      "userAgent": "agent"
    },
    "requestId": "id",
    "routeKey": "$default",
    "stage": "$default",
    "time": "12/Mar/2020:19:03:58 +0000",
    "timeEpoch": 1583348638390
  },
  "pathParameters": { "parameter1": "value1" },
  "stageVariables": { "stageVariable1": "value1", "stageVariable2": "value2" }
}
```

1.0

```
{
  "version": "1.0",
  "type": "REQUEST",
  "methodArn": "arn:aws:execute-api:us-east-1:123456789012:abcdef123/test/GET/
request",
  "identitySource": "user1,123",
  "authorizationToken": "user1,123",
  "resource": "/request",
  "path": "/request",
```

```
  "httpMethod": "GET",
  "headers": {
    "X-AMZ-Date": "20170718T062915Z",
    "Accept": "*/*",
    "HeaderAuth1": "headerValue1",
    "CloudFront-Viewer-Country": "US",
    "CloudFront-Forwarded-Proto": "https",
    "CloudFront-Is-Tablet-Viewer": "false",
    "CloudFront-Is-Mobile-Viewer": "false",
    "User-Agent": "..."
  },
  "queryStringParameters": {
    "QueryString1": "queryValue1"
  },
  "pathParameters": {},
  "stageVariables": {
    "StageVar1": "stageValue1"
  },
  "requestContext": {
    "path": "/request",
    "accountId": "123456789012",
    "resourceId": "05c7jb",
    "stage": "test",
    "requestId": "...",
    "identity": {
      "apiKey": "...",
      "sourceIp": "...",
      "clientCert": {
        "clientCertPem": "CERT_CONTENT",
        "subjectDN": "www.example.com",
        "issuerDN": "Example issuer",
        "serialNumber": "a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1",
        "validity": {
          "notBefore": "May 28 12:30:02 2019 GMT",
          "notAfter": "Aug  5 09:36:04 2021 GMT"
        }
      }
    },
    "resourcePath": "/request",
    "httpMethod": "GET",
    "apiId": "abcdef123"
  }
}
```

Formato della risposta dell'autorizzazione

Il tipo di formato del payload determina anche la struttura della risposta che deve essere restituita dalla funzione Lambda.

Risposta della funzione Lambda per il formato 1.0

Se scegli la versione del `1.0` formato, gli autorizzatori Lambda devono restituire una IAM politica che consenta o neghi l'accesso al percorso. API È possibile utilizzare la sintassi standard IAM della policy nella policy. Per esempi di policy IAM, consulta the section called “Controlla l'accesso per invocare un API”. È possibile passare le proprietà del contesto alle integrazioni Lambda o ai log di accesso utilizzando `$context.authorizer.`*`property`*. L'oggetto `context` è facoltativo e `claims` è un segnaposto riservato che non può essere utilizzato come oggetto contesto. Per ulteriori informazioni, consulta the section called “Variabili di logging”.

Example

```
{
  "principalId": "abcdef", // The principal user identification associated with the
 token sent by the client.
  "policyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": "execute-api:Invoke",
        "Effect": "Allow|Deny",
        "Resource": "arn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/
{httpVerb}/[{resource}/[{child-resources}]]"
      }
    ]
  },
  "context": {
    "exampleKey": "exampleValue"
  }
}
```

Risposta della funzione Lambda per il formato 2.0

Se scegli la versione del `2.0` formato, puoi restituire un valore booleano o un IAM criterio che utilizza la sintassi standard dei IAM criteri dalla tua funzione Lambda. Per restituire un valore booleano, abilitare risposte semplici per l'autorizzazione. Negli esempi seguenti viene illustrato il formato in cui

è necessario codificare il risultato restituito dalla funzione Lambda. L'oggetto `context` è facoltativo. È possibile passare le proprietà del contesto alle integrazioni Lambda o ai log di accesso utilizzando `$context.authorizer.property`. Per ulteriori informazioni, consulta the section called "Variabili di logging".

Simple response

```
{
  "isAuthorized": true/false,
  "context": {
    "exampleKey": "exampleValue"
  }
}
```

IAM policy

```
{
  "principalId": "abcdef", // The principal user identification associated with the
 token sent by the client.
  "policyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": "execute-api:Invoke",
        "Effect": "Allow|Deny",
        "Resource": "arn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/
{httpVerb}/[{resource}/[{child-resources}]]"
      }
    ]
  },
  "context": {
    "exampleKey": "exampleValue"
  }
}
```

Esempio di funzioni di autorizzazione Lambda

Le seguenti funzioni Lambda Node.js di esempio illustrano i formati di risposta richiesti che è necessario restituire dalla funzione Lambda per il tipo di formato del payload `2.0`.

Simple response - Node.js

```
export const handler = async(event) => {
    let response = {
        "isAuthorized": false,
        "context": {
            "stringKey": "value",
            "numberKey": 1,
            "booleanKey": true,
            "arrayKey": ["value1", "value2"],
            "mapKey": {"value1": "value2"}
        }
    };

    if (event.headers.authorization === "secretToken") {
        console.log("allowed");
        response = {
            "isAuthorized": true,
            "context": {
                "stringKey": "value",
                "numberKey": 1,
                "booleanKey": true,
                "arrayKey": ["value1", "value2"],
                "mapKey": {"value1": "value2"}
            }
        };
    }

    return response;

};
```

Simple response - Python

```
import json


def lambda_handler(event, context):
    response = {
        "isAuthorized": False,
        "context": {
            "stringKey": "value",
            "numberKey": 1,
```

```
            "booleanKey": True,
            "arrayKey": ["value1", "value2"],
            "mapKey": {"value1": "value2"}
        }
    }

    try:
        if (event["headers"]["authorization"] == "secretToken"):
            response = {
                "isAuthorized": True,
                "context": {
                    "stringKey": "value",
                    "numberKey": 1,
                    "booleanKey": True,
                    "arrayKey": ["value1", "value2"],
                    "mapKey": {"value1": "value2"}
                }
            }
            print('allowed')
            return response
        else:
            print('denied')
            return response
    except BaseException:
        print('denied')
        return response
```

IAM policy - Node.js

```
export const handler = async(event) => {
  if (event.headers.authorization == "secretToken") {
    console.log("allowed");
    return {
      "principalId": "abcdef", // The principal user identification associated with
 the token sent by the client.
      "policyDocument": {
        "Version": "2012-10-17",
        "Statement": [{
          "Action": "execute-api:Invoke",
          "Effect": "Allow",
          "Resource": event.routeArn
        }]
      },
```

```
      "context": {
        "stringKey": "value",
        "numberKey": 1,
        "booleanKey": true,
        "arrayKey": ["value1", "value2"],
        "mapKey": { "value1": "value2" }
      }
    };
  }
  else {
    console.log("denied");
    return {
      "principalId": "abcdef", // The principal user identification associated with
 the token sent by the client.
      "policyDocument": {
        "Version": "2012-10-17",
        "Statement": [{
          "Action": "execute-api:Invoke",
          "Effect": "Deny",
          "Resource": event.routeArn
        }]
      },
      "context": {
        "stringKey": "value",
        "numberKey": 1,
        "booleanKey": true,
        "arrayKey": ["value1", "value2"],
        "mapKey": { "value1": "value2" }
      }
    };
  }
};
```

IAM policy - Python

```
import json


def lambda_handler(event, context):
    response = {
        # The principal user identification associated with the token sent by
        # the client.
        "principalId": "abcdef",
```

```
        "policyDocument": {
            "Version": "2012-10-17",
            "Statement": [{
                "Action": "execute-api:Invoke",
                "Effect": "Deny",
                "Resource": event["routeArn"]
            }]
        },
        "context": {
            "stringKey": "value",
            "numberKey": 1,
            "booleanKey": True,
            "arrayKey": ["value1", "value2"],
            "mapKey": {"value1": "value2"}
        }
    }

    try:
        if (event["headers"]["authorization"] == "secretToken"):
            response = {
                # The principal user identification associated with the token
                # sent by the client.
                "principalId": "abcdef",
                "policyDocument": {
                    "Version": "2012-10-17",
                    "Statement": [{
                        "Action": "execute-api:Invoke",
                        "Effect": "Allow",
                        "Resource": event["routeArn"]
                    }]
                },
                "context": {
                    "stringKey": "value",
                    "numberKey": 1,
                    "booleanKey": True,
                    "arrayKey": ["value1", "value2"],
                    "mapKey": {"value1": "value2"}
                }
            }
            print('allowed')
            return response
        else:
            print('denied')
            return response
```

```
    except BaseException:
        print('denied')
        return response
```

Origini di identità

È facoltativamente possibile specificare le origini di identità per un'autorizzazione Lambda. Le origini di identità specificano la posizione dei dati necessari per autorizzare una richiesta. Ad esempio, è possibile specificare i valori di intestazione o di stringa di query come origini di identità. Se si specificano le origini di identità, i client devono includerle nella richiesta. Se la richiesta del client non include le fonti di identità, API Gateway non richiama l'autorizzatore Lambda e il client riceve un errore. `401`

La tabella seguente descrive le fonti di identità supportate per un autorizzatore Lambda.

Type	Esempio	Note
Valore intestazione	$request.header.*name*	I nomi delle intestazioni non fanno distinzione tra maiuscole e minuscole.
Valore della stringa di query	$ request.querystring.*name*	I nomi delle stringhe di query fanno distinzione tra maiuscole e minuscole.
Variabile di contesto	$ contesto.*variableName*	Valore di una variabile di contesto supportata.
Variabile di fase	$stageVariables.*variableName*	Il valore di una variabile di fase.

Caching delle risposte delle autorizzazioni

Puoi abilitare la memorizzazione nella cache per un autorizzatore Lambda specificando un. authorizerResultTtlInSeconds Quando la memorizzazione nella cache è abilitata per un autorizzatore, API Gateway utilizza le fonti di identità dell'autorizzatore come chiave della cache. Se un client specifica gli stessi parametri nelle fonti di identità all'interno del file configuratoTTL, API Gateway

utilizza il risultato dell'autorizzazione memorizzato nella cache, anziché richiamare la funzione Lambda.

Per abilitare il caching, l'autorizzazione deve disporre di almeno un'origine di identità.

Se abiliti le risposte semplici per un autorizzatore, la risposta dell'autorizzatore consente o nega completamente tutte le API richieste che corrispondono ai valori di origine dell'identità memorizzati nella cache. Per autorizzazioni più granulari, disabilita le risposte semplici e restituisci una policy. IAM

Per impostazione predefinita, API Gateway utilizza la risposta dell'autorizzatore memorizzata nella cache per tutte le rotte di un utente API che utilizza l'autorizzatore. Per inserire nella cache le risposte per l'instradamento, aggiungere `$context.routeKey` alle origini di identità dell'autorizzazione.

Creare un'autorizzazione Lambda

Quando si crea un autorizzatore Lambda, si specifica la funzione Lambda da utilizzare per Gateway. API È necessario concedere a API Gateway l'autorizzazione a richiamare la funzione Lambda utilizzando la politica delle risorse della funzione o IAM un ruolo. Per questo esempio, aggiorniamo la politica delle risorse per la funzione in modo che conceda a API Gateway l'autorizzazione a richiamare la nostra funzione Lambda.

```
aws apigatewayv2 create-authorizer \
    --api-id abcdef123 \
    --authorizer-type REQUEST \
    --identity-source '$request.header.Authorization' \
    --name lambda-authorizer \
    --authorizer-uri 'arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-west-2:123456789012:function:my-function/invocations' \
    --authorizer-payload-format-version '2.0' \
    --enable-simple-responses
```

Il comando seguente concede a API Gateway l'autorizzazione a richiamare la funzione Lambda. Se API Gateway non dispone dell'autorizzazione per richiamare la funzione, i client ricevono un. `500 Internal Server Error`

```
aws lambda add-permission \
    --function-name my-authorizer-function \
    --statement-id apigateway-invoke-permissions-abc123 \
    --action lambda:InvokeFunction \
    --principal apigateway.amazonaws.com \
```

```
    --source-arn "arn:aws:execute-api:us-west-2:123456789012:api-
id/authorizers/authorizer-id"
```

Dopo aver creato un autorizzatore e concesso a API Gateway l'autorizzazione a richiamarlo, aggiorna il percorso per utilizzare l'autorizzatore.

```
aws apigatewayv2 update-route \
    --api-id abcdef123 \
    --route-id acd123 \
    --authorization-type CUSTOM \
    --authorizer-id def123
```

Risoluzione dei problemi relativi alle autorizzazioni Lambda

Se API Gateway non può richiamare l'autorizzatore Lambda o l'autorizzatore Lambda restituisce una risposta in un formato non valido, i client ricevono un. `500 Internal Server Error`

Per risolvere gli errori, abilita la registrazione degli accessi per lo stage. API Includere la variabile di registrazione `$context.authorizer.error` nel formato di log.

Se i log indicano che API Gateway non è autorizzato a richiamare la tua funzione, aggiorna la politica delle risorse della funzione o fornisci un IAM ruolo per concedere a API Gateway l'autorizzazione a richiamare l'autorizzazione.

Se i log indicano che la funzione Lambda restituisce una risposta non valida, verificare che la funzione Lambda restituisca una risposta nel formato richiesto.

Controlla l'accesso HTTP APIs con gli JWT autorizzatori in API Gateway

Puoi usare JSON Web Tokens (JWTs) come parte dei framework OpenID OIDC Connect (OAuth) e 2.0 per limitare l'accesso dei client al tuo. APIs

Se configurate un sistema di JWT autorizzazione per un percorso del vostro computerAPI, API Gateway convalida quello JWTs che i client inviano con le richieste. API APIGateway consente o rifiuta le richieste in base alla convalida del token e, facoltativamente, agli ambiti del token. Se si configurano gli ambiti per un percorso, il token deve includere almeno uno degli ambiti del percorso.

È possibile configurare autorizzatori distinti per ogni percorso di un API percorso o utilizzare lo stesso autorizzatore per più percorsi.

> **Note**
>
> Non esiste un meccanismo standard per differenziare i token di JWT accesso da altri tipi di tokenJWTs, come i token ID OpenID Connect. A meno che non siano necessari token ID per API l'autorizzazione, si consiglia di configurare i percorsi in modo da richiedere gli ambiti di autorizzazione. Puoi anche configurare i tuoi JWT autorizzatori per richiedere gli emittenti o i destinatari che il tuo provider di identità utilizza solo per l'emissione di token di accesso. JWT

Autorizzazione delle richieste con un autorizzatore API JWT

APIGateway utilizza il seguente flusso di lavoro generale per autorizzare le richieste alle route configurate per utilizzare un autorizzatore. JWT

1. Controllare la presenza di `identitySource` per un token. Il `identitySource` può includere solo il token o il token con il prefisso `Bearer`.
2. Decodifica del token.
3. Controlla l'algoritmo e la firma del token utilizzando la chiave pubblica recuperata dal dell'emittent `jwks_uri`. Attualmente, sono supportati solo algoritmi RSA basati. APIGateway può memorizzare nella cache la chiave pubblica per due ore. Come best practice, quando si esegue la rotazione delle chiavi, definire un periodo di tolleranza durante il quale sia la vecchia che la nuova chiave sono valide.
4. Convalida delle richieste. APIGateway valuta le seguenti dichiarazioni relative ai token:
   - `kid`: il token deve presentare una richiesta di intestazione che corrisponde alla chiave nel `jwks_uri` che ha firmato il token.
   - `iss`: deve corrispondere all'`issuer` configurato per l'autorizzazione.
   - `aud` o `client_id`: devono corrispondere a una delle voci `audience` configurate per l'autorizzazione. APIIl gateway viene convalidato `client_id` solo se non `aud` è presente. Quando entrambi `client_id` sono `aud` presenti, API Gateway valutaaud.
   - `exp`— Deve essere successiva all'ora corrente di ingresso. UTC
   - `nbf`— Deve essere antecedente all'ora corrente di ingressoUTC.
   - `iat`— Deve essere antecedente all'ora corrente di arrivoUTC.
   - `scope` o `scp`: il token deve includere almeno uno degli ambiti negli `authorizationScopes` della route.

Se uno di questi passaggi fallisce, API Gateway nega la API richiesta.

Dopo aver convalidato ilJWT, API Gateway passa le attestazioni contenute nel token all'integrazione del API percorso. Le risorse di backend, come le funzioni Lambda, possono accedere alle JWT attestazioni. Ad esempio, se JWT include un'attestazione di identità`emailID`, è disponibile per un'integrazione Lambda in. `$event.requestContext.authorizer.jwt.claims.emailID` Per ulteriori informazioni sul payload che API Gateway invia alle integrazioni Lambda, consulta. the section called “AWS Lambda integrazioni”

Crea un autorizzatore JWT

Prima di creare un JWT autorizzatore, è necessario registrare un'applicazione client presso un provider di identità. È inoltre necessario aver creato un HTTPAPI. Per esempi di creazione di un HTTPAPI, vediCrea un'API HTTP.

Crea un JWT autorizzatore utilizzando la console

I passaggi seguenti mostrano come creare un JWT autorizzatore utilizzando la console.

Per creare un JWT autorizzatore utilizzando la console

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli un HTTPAPI.
3. Nel riquadro di navigazione principale, scegli Autorizzazione.
4. Scegli la scheda Gestisci gli autorizzatori.
5. Scegli Crea.
6. Per Tipo di autorizzazione, scegli. JWT
7. Configura il tuo JWT autorizzatore e specifica una fonte di identità che definisca l'origine del token.
8. Scegli Crea.

Crea un JWT autorizzatore utilizzando il AWS CLI

Il AWS CLI comando seguente crea un JWT autorizzatore. Per `jwt-configuration`, specificare `Audience` e `Issuer` per il provider di identità. Se utilizzi Amazon Cognito come provider di identità, lo è`IssuerUrl`. `https://cognito-idp.`*`us-east-2`*`.amazonaws.com/`*`userPoolID`*

```
aws apigatewayv2 create-authorizer \
    --name authorizer-name \
```

```
    --api-id api-id \
    --authorizer-type JWT \
    --identity-source '$request.header.Authorization' \
    --jwt-configuration Audience=audience,Issuer=IssuerUrl
```

Crea un JWT autorizzatore utilizzando AWS CloudFormation

Il AWS CloudFormation modello seguente crea un account HTTP API con un JWT autorizzatore che utilizza Amazon Cognito come provider di identità.

L'output del AWS CloudFormation modello è URL per un'interfaccia utente ospitata da Amazon Cognito in cui i clienti possono registrarsi e accedere per ricevere un. JWT Dopo l'accesso, il cliente viene reindirizzato al tuo account HTTP API con un token di accesso nel. URL Per richiamare il API con il token di accesso, modifica il # in in URL ? a per utilizzare il token come parametro della stringa di query.

Modello di esempio AWS CloudFormation

```
AWSTemplateFormatVersion: '2010-09-09'
Description: |
  Example HTTP API with a JWT authorizer. This template includes an Amazon Cognito user
 pool as the issuer for the JWT authorizer
  and an Amazon Cognito app client as the audience for the authorizer. The outputs
 include a URL for an Amazon Cognito hosted UI where clients can
  sign up and sign in to receive a JWT. After a client signs in, the client is
 redirected to your HTTP API with an access token
  in the URL. To invoke the API with the access token, change the '#' in the URL to a
 '?' to use the token as a query string parameter.

Resources:
  MyAPI:
    Type: AWS::ApiGatewayV2::Api
    Properties:
      Description: Example HTTP API
      Name: api-with-auth
      ProtocolType: HTTP
      Target: !GetAtt MyLambdaFunction.Arn
  DefaultRouteOverrides:
    Type: AWS::ApiGatewayV2::ApiGatewayManagedOverrides
    Properties:
      ApiId: !Ref MyAPI
      Route:
        AuthorizationType: JWT
```

```
        AuthorizerId: !Ref JWTAuthorizer
  JWTAuthorizer:
    Type: AWS::ApiGatewayV2::Authorizer
    Properties:
      ApiId: !Ref MyAPI
      AuthorizerType: JWT
      IdentitySource:
        - '$request.querystring.access_token'
      JwtConfiguration:
        Audience:
        - !Ref AppClient
        Issuer: !Sub https://cognito-idp.${AWS::Region}.amazonaws.com/${UserPool}
      Name: test-jwt-authorizer
  MyLambdaFunction:
    Type: AWS::Lambda::Function
    Properties:
      Runtime: nodejs18.x
      Role: !GetAtt FunctionExecutionRole.Arn
      Handler: index.handler
      Code:
        ZipFile: |
          exports.handler = async (event) => {
              const response = {
                  statusCode: 200,
                  body: JSON.stringify('Hello from the ' + event.routeKey + ' route!'),
              };
              return response;
          };
  APIInvokeLambdaPermission:
    Type: AWS::Lambda::Permission
    Properties:
      FunctionName: !Ref MyLambdaFunction
      Action: lambda:InvokeFunction
      Principal: apigateway.amazonaws.com
      SourceArn: !Sub arn:${AWS::Partition}:execute-api:${AWS::Region}:
${AWS::AccountId}:${MyAPI}/$default/$default
  FunctionExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
```

```
            Service:
            - lambda.amazonaws.com
          Action:
            - 'sts:AssumeRole'
    ManagedPolicyArns:
      - arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole
  UserPool:
    Type: AWS::Cognito::UserPool
    Properties:
      UserPoolName: http-api-user-pool
      AutoVerifiedAttributes:
        - email
      Schema:
        - Name: name
          AttributeDataType: String
          Mutable: true
          Required: true
        - Name: email
          AttributeDataType: String
          Mutable: false
          Required: true
  AppClient:
    Type: AWS::Cognito::UserPoolClient
    Properties:
      AllowedOAuthFlows:
        - implicit
      AllowedOAuthScopes:
        - aws.cognito.signin.user.admin
        - email
        - openid
        - profile
      AllowedOAuthFlowsUserPoolClient: true
      ClientName: api-app-client
      CallbackURLs:
        - !Sub https://${MyAPI}.execute-api.${AWS::Region}.amazonaws.com
      ExplicitAuthFlows:
        - ALLOW_USER_PASSWORD_AUTH
        - ALLOW_REFRESH_TOKEN_AUTH
      UserPoolId: !Ref UserPool
      SupportedIdentityProviders:
        - COGNITO
  HostedUI:
    Type: AWS::Cognito::UserPoolDomain
    Properties:
```

```
      Domain: !Join
        - '-'
        - - !Ref MyAPI
          - !Ref AppClient
      UserPoolId: !Ref UserPool
Outputs:
  SignupURL:
    Value: !Sub https://${HostedUI}.auth.${AWS::Region}.amazoncognito.com/login?
client_id=${AppClient}&response_type=token&scope=email+profile&redirect_uri=https://
${MyAPI}.execute-api.${AWS::Region}.amazonaws.com
```

Aggiorna un percorso per utilizzare un JWT autorizzatore

È possibile utilizzare la console AWS CLI, o un AWS SDK per aggiornare un percorso per utilizzare un JWT autorizzatore.

Aggiorna un percorso per utilizzare un JWT autorizzatore utilizzando la console

I passaggi seguenti mostrano come aggiornare un percorso per utilizzare l'JWTautorizzatore utilizzando la console.

Per creare un JWT autorizzatore utilizzando la console

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli un HTTPAPI.
3. Nel riquadro di navigazione principale, scegli Autorizzazione.
4. Scegli un metodo, quindi seleziona l'autorizzatore dal menu a discesa e scegli Allega l'autorizzatore.

Aggiorna un percorso per utilizzare un JWT autorizzatore utilizzando il AWS CLI

Il comando seguente aggiorna una route per utilizzare un JWT autorizzatore utilizzando. AWS CLI

```
aws apigatewayv2 update-route \
   --api-id api-id  \
   --route-id route-id  \
   --authorization-type JWT \
   --authorizer-id authorizer-id \
   --authorization-scopes user.email
```

Controlla l'accesso HTTP APIs con IAM autorizzazione in API Gateway

È possibile abilitare IAM l'autorizzazione per i HTTP API percorsi. Quando IAM l'autorizzazione è abilitata, i client devono utilizzare Signature Version 4 (SigV4) per firmare le proprie richieste con AWS le credenziali. APIGateway richiama il API percorso solo se il client dispone `execute-api` dell'autorizzazione per il percorso.

IAMl'autorizzazione per HTTP APIs è simile a quella per. RESTAPIs

> **Note**
>
> Le politiche relative alle risorse non sono attualmente supportate per HTTPAPIs.

Per esempi di IAM politiche che concedono ai client il permesso di invocazioneAPIs, consultathe section called “Controlla l'accesso per invocare un API”.

Abilitazione dell'autorizzazione IAM per un instradamento

Il AWS CLI comando seguente abilita IAM l'autorizzazione per una HTTP API route.

```
aws apigatewayv2 update-route \
    --api-id abc123 \
    --route-id abcdef \
    --authorization-type AWS_IAM
```

Crea integrazioni per le API HTTP in API Gateway

Le integrazioni collegano una route alle risorse di back-end. Le API HTTP supportano le integrazioni di proxy Lambda AWS , servizio e proxy HTTP. Ad esempio, è possibile configurare una richiesta `POST` per la route `/signup` dell'API per l'integrazione con una funzione Lambda che gestisce la registrazione dei clienti.

Argomenti

- Crea integrazioni AWS Lambda proxy per HTTP APIs in API Gateway
- Crea integrazioni proxy HTTP per API HTTP
- Crea integrazioni AWS di servizi per HTTP APIs in API Gateway
- Crea integrazioni private per HTTP APIs in API Gateway

Crea integrazioni AWS Lambda proxy per HTTP APIs in API Gateway

L'integrazione del proxy Lambda consente di integrare un API percorso con una funzione Lambda. Quando un client chiama il tuoAPI, API Gateway invia la richiesta alla funzione Lambda e restituisce la risposta della funzione al client. Per esempi di creazione di un HTTPAPI, vediCrea un'API HTTP.

Tipo di formato payload

La versione del formato payload specifica il formato dell'evento che API Gateway invia a un'integrazione Lambda e il modo in cui API Gateway interpreta la risposta da Lambda. Se non si specifica una versione del formato di payload, AWS Management Console utilizza la versione più recente per impostazione predefinita. Se crei un'integrazione Lambda utilizzando AWS CLI, AWS CloudFormation o anSDK, devi specificare a. `payloadFormatVersion` I valori supportati sono `1.0` e `2.0`.

Per ulteriori informazioni su come impostare il`payloadFormatVersion`, consulta create-integration. Per ulteriori informazioni su come determinare il livello `payloadFormatVersion` di un'integrazione esistente, vedi get-integration

Differenze nel formato del payload

L'elenco seguente mostra le differenze tra le versioni del formato `1.0` e del `2.0` payload:

- Il formato `2.0` non contiene i campi `multiValueHeaders` o `multiValueQueryStringParameters`. Le intestazioni duplicate sono combinate con virgole e incluse nel campo `headers`. Le stringhe di query duplicate sono combinate con virgole e incluse nel campo `queryStringParameters`.
- Il formato `2.0` ha`rawPath`. Se utilizzi una API mappatura per connettere il tuo stage a un nome di dominio personalizzato, `rawPath` non fornirà il valore di API mappatura. Usa format `1.0` e `path` per accedere alla API mappatura del tuo nome di dominio personalizzato.
- Il formato `2.0` include un nuovo campo `cookies`. Tutte le intestazioni dei cookie nella richiesta vengono combinate con virgole e aggiunte al campo `cookies`. Nella risposta al client, ogni cookie diventa un'intestazione `set-cookie`.

Struttura del formato del payload

Gli esempi seguenti mostrano la struttura di ogni tipo di formato payload. Tutti i nomi delle intestazioni sono in minuscolo.

2.0

```
{
  "version": "2.0",
  "routeKey": "$default",
  "rawPath": "/my/path",
  "rawQueryString": "parameter1=value1&parameter1=value2&parameter2=value",
  "cookies": [
    "cookie1",
    "cookie2"
  ],
  "headers": {
    "header1": "value1",
    "header2": "value1,value2"
  },
  "queryStringParameters": {
    "parameter1": "value1,value2",
    "parameter2": "value"
  },
  "requestContext": {
    "accountId": "123456789012",
    "apiId": "api-id",
    "authentication": {
      "clientCert": {
        "clientCertPem": "CERT_CONTENT",
        "subjectDN": "www.example.com",
        "issuerDN": "Example issuer",
        "serialNumber": "a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1",
        "validity": {
          "notBefore": "May 28 12:30:02 2019 GMT",
          "notAfter": "Aug  5 09:36:04 2021 GMT"
        }
      }
    },
    "authorizer": {
      "jwt": {
        "claims": {
          "claim1": "value1",
          "claim2": "value2"
        },
        "scopes": [
          "scope1",
          "scope2"
        ]
```

```
      }
    },
    "domainName": "id.execute-api.us-east-1.amazonaws.com",
    "domainPrefix": "id",
    "http": {
      "method": "POST",
      "path": "/my/path",
      "protocol": "HTTP/1.1",
      "sourceIp": "192.0.2.1",
      "userAgent": "agent"
    },
    "requestId": "id",
    "routeKey": "$default",
    "stage": "$default",
    "time": "12/Mar/2020:19:03:58 +0000",
    "timeEpoch": 1583348638390
  },
  "body": "Hello from Lambda",
  "pathParameters": {
    "parameter1": "value1"
  },
  "isBase64Encoded": false,
  "stageVariables": {
    "stageVariable1": "value1",
    "stageVariable2": "value2"
  }
}
```

1.0

```
{
  "version": "1.0",
  "resource": "/my/path",
  "path": "/my/path",
  "httpMethod": "GET",
  "headers": {
    "header1": "value1",
    "header2": "value2"
  },
  "multiValueHeaders": {
    "header1": [
      "value1"
    ],
```

```
    "header2": [
      "value1",
      "value2"
    ]
  },
  "queryStringParameters": {
    "parameter1": "value1",
    "parameter2": "value"
  },
  "multiValueQueryStringParameters": {
    "parameter1": [
      "value1",
      "value2"
    ],
    "parameter2": [
      "value"
    ]
  },
  "requestContext": {
    "accountId": "123456789012",
    "apiId": "id",
    "authorizer": {
      "claims": null,
      "scopes": null
    },
    "domainName": "id.execute-api.us-east-1.amazonaws.com",
    "domainPrefix": "id",
    "extendedRequestId": "request-id",
    "httpMethod": "GET",
    "identity": {
      "accessKey": null,
      "accountId": null,
      "caller": null,
      "cognitoAuthenticationProvider": null,
      "cognitoAuthenticationType": null,
      "cognitoIdentityId": null,
      "cognitoIdentityPoolId": null,
      "principalOrgId": null,
      "sourceIp": "192.0.2.1",
      "user": null,
      "userAgent": "user-agent",
      "userArn": null,
      "clientCert": {
        "clientCertPem": "CERT_CONTENT",
```

```
        "subjectDN": "www.example.com",
        "issuerDN": "Example issuer",
        "serialNumber": "a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1",
        "validity": {
          "notBefore": "May 28 12:30:02 2019 GMT",
          "notAfter": "Aug  5 09:36:04 2021 GMT"
        }
      }
    },
    "path": "/my/path",
    "protocol": "HTTP/1.1",
    "requestId": "id=",
    "requestTime": "04/Mar/2020:19:15:17 +0000",
    "requestTimeEpoch": 1583349317135,
    "resourceId": null,
    "resourcePath": "/my/path",
    "stage": "$default"
  },
  "pathParameters": null,
  "stageVariables": null,
  "body": "Hello from Lambda!",
  "isBase64Encoded": false
}
```

Formato di risposta della funzione Lambda

Il tipo di formato del payload determina la struttura della risposta che deve essere restituita dalla funzione Lambda.

Risposta della funzione Lambda per il formato 1.0

Con la versione `1.0` in formato, le integrazioni Lambda devono restituire una risposta nel seguente formato: JSON

Example

```
{
    "isBase64Encoded": true|false,
    "statusCode": httpStatusCode,
    "headers": { "headername": "headervalue", ... },
    "multiValueHeaders": { "headername": ["headervalue", "headervalue2", ...], ... },
    "body": "..."
```

```
}
```

Risposta della funzione Lambda per il formato 2.0

Con la versione `2.0` in formato, API Gateway può dedurre automaticamente il formato della risposta. APIGateway fa i seguenti presupposti se la funzione Lambda torna JSON valida e non restituisce un: `statusCode`

- `isBase64Encoded` è `false`.
- `statusCode` è `200`.
- `content-type` è `application/json`.
- `body` è la risposta della funzione.

Gli esempi seguenti mostrano l'output di una funzione Lambda e l'interpretazione di API Gateway.

Risultato della funzione Lambda	APIInterpretazione del gateway
`"Hello from Lambda!"`	`{ "isBase64Encoded": false, "statusCode": 200, "body": "Hello from Lambda!", "headers": { "content-type": "application/json" } }`
`{ "message": "Hello from Lambda!" }`	`{ "isBase64Encoded": false, "statusCode": 200, "body": "{ \"message\": \"Hello from Lambda!\" }", "headers": { "content-type": "application/json" } }`

Per personalizzare la risposta, la funzione Lambda deve restituire una risposta con il seguente formato.

```
{
    "cookies" : ["cookie1", "cookie2"],
    "isBase64Encoded": true|false,
    "statusCode": httpStatusCode,
    "headers": { "headername": "headervalue", ... },
    "body": "Hello from Lambda!"
}
```

Crea integrazioni proxy HTTP per API HTTP

Un'integrazione proxy HTTP consente di collegare una route API a un endpoint HTTP instradabile pubblicamente. Con questo tipo di integrazione, API Gateway passa l'intera richiesta e risposta tra il front-end e il back-end.

Per creare un'integrazione HTTP, fornire l'URL di un endpoint HTTP instradabile pubblicamente.

Integrazione proxy HTTP con variabili di percorso

Nelle route delle API HTTP è possibile utilizzare delle variabili di percorso.

Ad esempio, la route `/pets/{petID}` cattura le richieste a `/pets/6`. Puoi fare riferimento alle variabili di percorso nell'URI di integrazione per inviare il contenuto della variabile a un'integrazione. Un esempio è `/pets/extendedpath/{petID}`.

Puoi utilizzare variabili di percorso greedy per catturare tutte le risorse figlio di una route. Per creare una variabile di percorso greedy, aggiungere + al nome della variale, ad esempio `{proxy+}`.

Per impostare una route con un'integrazione proxy HTTP che cattura tutte le richieste, crea una route API con una variabile di percorso greedy (ad esempio, `/parent/{proxy+}`). Integra la route con un endpoint HTTP (ad esempio `https://petstore-demo-endpoint.execute-api.com/petstore/{proxy}`) nel metodo `ANY`. La variabile di percorso greedy deve trovarsi alla fine del percorso della risorsa.

Crea integrazioni AWS di servizi per HTTP APIs in API Gateway

Puoi integrare il tuo HTTP API con i AWS servizi utilizzando integrazioni di prima classe. Un'integrazione di prima classe collega un HTTP API percorso a un servizio. AWS API Quando un client richiama un percorso supportato da un'integrazione di prima classe, API Gateway richiama

un servizio per te. AWS API Ad esempio, puoi utilizzare integrazioni di prima classe per inviare un messaggio a una coda di Amazon Simple Queue Service o per avviare una macchina a stati. AWS Step Functions Per le operazioni di servizio supportate, consulta the section called “AWS riferimento alle integrazioni di servizi”.

Mappatura dei parametri delle richieste

Le integrazioni di prima classe hanno parametri obbligatori e facoltativi. È necessario configurare tutti i parametri obbligatori per creare un'integrazione. È possibile utilizzare valori statici o mappare i parametri che vengono valutati dinamicamente in fase di runtime. Per un elenco completo delle integrazioni e dei parametri supportati, consulta the section called “AWS riferimento alle integrazioni di servizi”.

La tabella seguente descrive i parametri di richiesta di mappatura supportati.

Type	Esempio	Note
Valore intestazione	$request.header.*name*	I nomi delle intestazioni non fanno distinzione tra maiuscole e minuscole. APIAd esempio, Gateway combina più valori di intestazione con virgole. `"header1": "value1,value2"`
Valore della stringa di query	$request.querystring.*name*	I nomi delle stringhe di query fanno distinzione tra maiuscole e minuscole. APIAd esempio, Gateway combina più valori con virgole. `"querystring1": "Value1,Value2"`
Parametro del percorso	$request.path.*name*	Il valore di un parametro del percorso nella richiesta . Ad esempio, se il percorso è`/pets/{petId}` , puoi mappare il `petId` parametro

Type	Esempio	Note
		della richiesta con *$request.path.petId* .
Passthrough corpo richiesta	$request.body	APIII gateway trasmette l'intero corpo della richiesta.
Corpo della richiesta	$request.body.*name*	Un'JSONespressione di percorso. Le espressioni di discesa ricorsiva (`$request.body..` *name*) e di filtro (?(*expression*)) non sono supportate. **Note** Quando si specifica un JSON percorso, API Gateway tronca il corpo della richiesta a 100 KB e quindi applica l'espressione di selezione. Per inviare payload superiori a 100 KB, specificare `$request.body` .
Variabile di contesto	$contesto.*variableName*	Valore di una variabile di contesto supportata.
Variabile di fase	$stageVariables.*variableName*	Il valore di una variabile di fase.
Valore statico	*string*	Un valore costante.

Creazione di un'integrazione di prima classe

Prima di creare un'integrazione di prima classe, è necessario creare un IAM ruolo che conceda a API Gateway le autorizzazioni per richiamare l'azione di AWS servizio con cui si sta effettuando l'integrazione. Per ulteriori informazioni, consulta Creazione di un ruolo per un servizio AWS.

Per creare un'integrazione di prima classe, scegli un'azione di AWS servizio supportata, ad esempio configura i parametri di richiesta e fornisci un ruolo che conceda a API Gateway le autorizzazioni per richiamare il servizio integrato. `SQS-SendMessage` AWS API A seconda del sottotipo di integrazione, sono necessari diversi parametri di richiesta. Per ulteriori informazioni, consulta the section called "AWS riferimento alle integrazioni di servizi".

Il AWS CLI comando seguente crea un'integrazione che invia un SQS messaggio Amazon.

```
aws apigatewayv2 create-integration \
    --api-id abcdef123 \
    --integration-subtype SQS-SendMessage \
    --integration-type AWS_PROXY \
    --payload-format-version 1.0 \
    --credentials-arn arn:aws:iam::123456789012:role/apigateway-sqs \
    --request-parameters '{"QueueUrl": "$request.header.queueUrl", "MessageBody":
 "$request.body.message"}'
```

Crea un'integrazione di prima classe utilizzando AWS CloudFormation

L'esempio seguente mostra uno AWS CloudFormation snippet che crea un `/{source}/{detailType}` percorso con un'integrazione di prima classe con Amazon. EventBridge

Il parametro `Source` viene mappato al parametro del percorso `{source}`, `DetailType` viene mappato al parametro del percorso `{DetailType}` e il parametro `Detail` viene mappato al corpo della richiesta.

Lo snippet non mostra il bus degli eventi o il IAM ruolo che concede a API Gateway le autorizzazioni per richiamare l'azione. `PutEvents`

```
Route:
    Type: AWS::ApiGatewayV2::Route
    Properties:
      ApiId: !Ref HttpApi
      AuthorizationType: None
      RouteKey: 'POST /{source}/{detailType}'
```

```
      Target: !Join
        - /
        - - integrations
          - !Ref Integration
  Integration:
    Type: AWS::ApiGatewayV2::Integration
    Properties:
      ApiId: !Ref HttpApi
      IntegrationType: AWS_PROXY
      IntegrationSubtype: EventBridge-PutEvents
      CredentialsArn: !GetAtt EventBridgeRole.Arn
      RequestParameters:
        Source: $request.path.source
        DetailType: $request.path.detailType
        Detail: $request.body
        EventBusName: !GetAtt EventBus.Arn
      PayloadFormatVersion: "1.0"
```

Riferimento al sottotipo di integrazione

Sono supportati i seguenti sottotipi di integrazione per. HTTP APIs

Sottotipi di integrazione

- EventBridge- 1.0 PutEvents
- SQS- SendMessage 1.0
- SQS- 1.0 ReceiveMessage
- SQS- 1.0 DeleteMessage
- SQS- 1.0 PurgeQueue
- AppConfig- 1.0 GetConfiguration
- Kinesis 1.0 PutRecord
- StepFunctions- 1.0 StartExecution
- StepFunctions- StartSyncExecution 1.0
- StepFunctions- StopExecution 1.0

EventBridge- 1.0 PutEvents

Invia eventi personalizzati ad Amazon EventBridge in modo che possano essere abbinati alle regole.

Parametro	Campo obbligatorio
Dettaglio	True
DetailType	True
Crea	True
Orario	False
EventBusName	False
Risorse	False
Regione	False
TraceHeader	False

Per ulteriori informazioni, PutEventsconsulta Amazon EventBridge API Reference.

SQS- SendMessage 1.0

Recapita un messaggio alla coda specificata.

Parametro	Obbligatorio
QueueUrl	True
MessageBody	True
DelaySeconds	False
MessageAttributes	False
MessageDeduplicationId	False
MessageGroupId	False
MessageSystemAttributes	False
Regione	False

Per ulteriori informazioni, consulta SendMessageAmazon Simple Queue Service API Reference.

SQS- 1.0 ReceiveMessage

Recupera uno o più messaggi (fino a 10) dalla coda specificata.

Parametro	Obbligatorio
QueueUrl	True
AttributeNames	False
MaxNumberOfMessages	False
MessageAttributeNames	False
ReceiveRequestAttemptId	False
VisibilityTimeout	False
WaitTimeSeconds	False
Regione	False

Per ulteriori informazioni, consulta ReceiveMessageAmazon Simple Queue Service API Reference.

SQS- 1.0 DeleteMessage

Elimina il messaggio specificato dalla coda specificata.

Parametro	Obbligatorio
ReceiptHandle	True
QueueUrl	True
Regione	False

Per ulteriori informazioni, consulta DeleteMessageAmazon Simple Queue Service API Reference.

SQS- 1.0 PurgeQueue

Elimina tutti i messaggi nella coda specificata.

Parametro	Obbligatorio
QueueUrl	True
Regione	False

Per ulteriori informazioni, consulta PurgeQueueAmazon Simple Queue Service API Reference.

AppConfig- 1.0 GetConfiguration

Ricezione di informazioni su una configurazione.

Parametro	Campo obbligatorio
Applicazione	True
Ambiente	True
Configurazione	True
ClientId	True
ClientConfigurationVersion	False
Regione	False

Per ulteriori informazioni, consulta la GetConfigurationsezione AWS AppConfig APIReference.

Kinesis 1.0 PutRecord

Scrive un singolo record di dati in un flusso di dati di Amazon Kinesis.

Parametro	Obbligatorio
StreamName	True

Parametro	Obbligatorio
Dati	True
PartitionKey	True
SequenceNumberForOrdering	False
ExplicitHashKey	False
Regione	False

Per ulteriori informazioni, consulta PutRecordAmazon Kinesis API Data Streams Reference.

StepFunctions- 1.0 StartExecution

Avvia l'esecuzione di una macchina a stati.

Parametro	Obbligatorio
StateMachineArn	True
Nome	False
Input	False
Regione	False

Per ulteriori informazioni, consulta la StartExecutionsezione AWS Step Functions APIReference.

StepFunctions- StartSyncExecution 1.0

Avvia un'esecuzione sincrona di una macchina a stati.

Parametro	Obbligatorio
StateMachineArn	True
Nome	False

Parametro	Obbligatorio
Input	False
Regione	False
TraceHeader	False

Per ulteriori informazioni, consulta la StartSyncExecutionsezione AWS Step Functions APIReference.

StepFunctions- StopExecution 1.0

Interrompe un'esecuzione.

Parametro	Obbligatorio
ExecutionArn	True
Causa	False
Errore	False
Regione	False

Per ulteriori informazioni, consulta la StopExecutionsezione AWS Step Functions APIReference.

Crea integrazioni private per HTTP APIs in API Gateway

Le integrazioni private consentono di creare API integrazioni con risorse private in un ambienteVPC, come Application Load Balancers o applicazioni basate su container AmazonECS.

Puoi esporre le tue risorse in modo che possano accedervi anche clienti esterni utilizzando integrazioni private. VPC VPC Puoi controllare l'accesso al tuo dispositivo API utilizzando uno qualsiasi dei metodi di autorizzazione supportati da API Gateway.

Per creare un'integrazione privata, devi prima creare un VPC link. Per ulteriori informazioni sui VPC link, consultaConfigurare VPC i collegamenti per HTTP APIs in API Gateway.

Dopo aver creato un VPC link, puoi configurare integrazioni private che si connettono a un Application Load Balancer, Network Load Balancer o a risorse registrate con un servizio. AWS Cloud Map

Per creare un'integrazione privata, tutte le risorse devono appartenere allo stesso AWS account (inclusi il sistema di bilanciamento del carico o il AWS Cloud Map servizio, VPC il link e). HTTP API

Per impostazione predefinita, il traffico di integrazione privato utilizza il HTTP protocollo. È possibile specificare a `tlsConfig`se è necessario utilizzare il traffico di integrazione privatoHTTPS. A tale scopo AWS Management Console, quando crei l'integrazione privata, scegli Impostazioni avanzate e quindi inserisci un nome server sicuro.

> **Note**
>
> Per le integrazioni private, API Gateway include la parte relativa alla fase dell'APIendpoint nella richiesta alle risorse di backend. Ad esempio, una richiesta allo `test` stadio di un API include `test/`*`route-path`* nella richiesta di integrazione privata. Per rimuovere il nome della fase dalla richiesta alle risorse di back-end, utilizzare parameter mapping (mappatura dei parametri) per sovrascrivere il percorso della richiesta a `$request.path`.

Creare un'integrazione privata utilizzando un Application Load Balancer o un Network Load Balancer

Prima di creare un'integrazione privata, devi creare un VPC link. Per ulteriori informazioni sui VPC link, consultaConfigurare VPC i collegamenti per HTTP APIs in API Gateway.

Per creare un'integrazione privata con un Application Load Balancer o Network Load Balancer, crea HTTP un'integrazione proxy, specifica VPC il link da utilizzare e fornisci il ARN listener del load balancer.

Utilizzate il comando seguente per creare un'integrazione privata che si connetta a un sistema di bilanciamento del carico tramite un collegamento. VPC

```
aws apigatewayv2 create-integration --api-id api-id --integration-type HTTP_PROXY \
    --integration-method GET --connection-type VPC_LINK \
    --connection-id VPC-link-ID \
    --integration-uri arn:aws:elasticloadbalancing:us-east-2:123456789012:listener/app/
my-load-balancer/50dc6c495c0c9188/0467ef3c8400ae65
    --payload-format-version 1.0
```

Crea un'integrazione privata utilizzando AWS Cloud Map Service Discovery

Prima di creare un'integrazione privata, devi creare un VPC link. Per ulteriori informazioni sui VPC link, consultaConfigurare VPC i collegamenti per HTTP APIs in API Gateway.

Per le integrazioni con AWS Cloud Map, API Gateway utilizza `DiscoverInstances` per identificare le risorse. È possibile utilizzare i parametri di query per fare riferimento a risorse specifiche. Gli attributi delle risorse registrate devono includere indirizzi IP e porte. APIGateway distribuisce le richieste tra risorse integre da cui vengono restituite. `DiscoverInstances` Per ulteriori informazioni, consulta DiscoverInstancesla sezione AWS Cloud Map API Reference.

> **Note**
>
> Se usi Amazon ECS per compilare le voci AWS Cloud Map, devi configurare la tua ECS attività Amazon per utilizzare i SRV record con Amazon ECS Service Discovery o attivare Amazon ECS Service Connect. Per ulteriori informazioni, consulta Interconnecting services nella Guida per gli sviluppatori di Amazon Elastic Container Service.

Per creare un'integrazione privata con AWS Cloud Map, crea un'integrazione HTTP proxy, specifica il VPC link da utilizzare e fornisci il ARN AWS Cloud Map servizio.

Utilizza il comando seguente per creare un'integrazione privata che utilizzi il rilevamento dei AWS Cloud Map servizi per identificare le risorse.

```
aws apigatewayv2 create-integration --api-id api-id --integration-type HTTP_PROXY  \
    --integration-method GET --connection-type VPC_LINK \
    --connection-id VPC-link-ID \
    --integration-uri arn:aws:servicediscovery:us-east-2:123456789012:service/srv-id?
stage=prod&deployment=green_deployment
    --payload-format-version 1.0
```

Configurare VPC i collegamenti per HTTP APIs in API Gateway

VPCi link ti consentono di creare integrazioni private che collegano i tuoi HTTP API percorsi a risorse private in un ambienteVPC, come Application Load Balancers o applicazioni basate su container AmazonECS. Per ulteriori informazioni sulla creazione di integrazioni private, consultare Crea integrazioni private per HTTP APIs in API Gateway.

Un'integrazione privata utilizza un VPC collegamento per incapsulare le connessioni tra Gateway e risorse mirate. API VPC È possibile riutilizzare VPC i collegamenti su percorsi diversi e. APIs

Quando crei un VPC collegamento, API Gateway crea e gestisce interfacce di rete elastiche per il VPC collegamento nel tuo account. Questo processo può richiedere alcuni minuti. Quando un VPC link è pronto per l'uso, il suo stato passa da a`PENDING`. `AVAILABLE`

> **Note**
>
> Se non viene inviato traffico tramite il VPC collegamento per 60 giorni, lo diventa`INACTIVE`. Quando un VPC collegamento è in uno `INACTIVE` stato, API Gateway elimina tutte le interfacce di rete del VPC collegamento. Ciò fa sì che API le richieste che dipendono dal VPC collegamento abbiano esito negativo. Se API le richieste riprendono, API Gateway esegue nuovamente il provisioning delle interfacce di rete. La creazione delle interfacce di rete e la riattivazione del collegamento possono richiedere alcuni minuti. VPC È possibile utilizzare lo stato del VPC collegamento per monitorare lo stato del collegamento. VPC

Crea un VPC link utilizzando il AWS CLI

Utilizzate il seguente comando per creare un VPC collegamento. Per creare un VPC collegamento, tutte le risorse coinvolte devono appartenere allo stesso AWS account.

```
aws apigatewayv2 create-vpc-link --name MyVpcLink \
    --subnet-ids subnet-aaaa subnet-bbbb \
    --security-group-ids sg1234 sg5678
```

> **Note**
>
> VPCi link sono immutabili. Dopo aver creato un VPC link, non è possibile modificarne le sottoreti o i gruppi di sicurezza.

Eliminare un VPC collegamento utilizzando il AWS CLI

Utilizzate il seguente comando per eliminare un VPC collegamento.

```
aws apigatewayv2 delete-vpc-link --vpc-link-id abcd123
```

Disponibilità in base alla Regione

VPCi collegamenti per HTTP APIs sono supportati nelle seguenti regioni e zone di disponibilità:

Nome Regione	Regione	Zone di disponibilità supportate
Stati Uniti orientali (Ohio)	us-east-2	use2-az1, use2-az2, use2-az3
Stati Uniti orientali (Virginia settentrionale)	us-east-1	use1-az1, use1-az2, use1-az4, use1-az5, use1-az6
US West (N. California)	us-west-1	usw1-az1, usw1-az3
Stati Uniti occidentali (Oregon)	us-west-2	usw2-az1, usw2-az2, usw2-az3, usw2-az4
Asia Pacifico (Hong Kong)	ap-east-1	ape1-az2, ape1-az3
Asia Pacific (Mumbai)	ap-south-1	aps1-az1, aps1-az2, aps1-az3
Asia Pacifico (Seul)	ap-northeast-2	apne2-az1, apne2-az2, apne2-az3
Asia Pacific (Singapore)	ap-southeast-1	apse1-az1, apse1-az2, apse1-az3
Asia Pacific (Sydney)	ap-southeast-2	apse2-az1, apse2-az2, apse2-az3

Nome Regione	Regione	Zone di disponibilità supportate
Asia Pacifico (Tokyo)	ap-northeast-1	apne1-az1, apne1-az2, apne1-az4
Canada (Central)	ca-central-1	cac1-az1, cac1-az2
Europe (Frankfurt)	eu-central-1	euc1-az1, euc1-az2, euc1-az3
Europa (Irlanda)	eu-west-1	euw1-az1, euw1-az2, euw1-az3
Europe (London)	eu-west-2	euw2-az1, euw2-az2, euw2-az3
Europe (Paris)	eu-west-3	euw3-az1, euw3-az3
Europa (Stoccolma)	eu-north-1	eun1-az1, eun1-az2, eun1-az3
Medio Oriente (Bahrein)	me-south-1	mes1-az1, mes1-az2, mes1-az3
Sud America (São Paulo)	sa-east-1	sae1-az1, sae1-az2, sae1-az3
AWS GovCloud (Stati Uniti occidentali)	us-gov-west-1	usgw1-az1, usgw1-az2, usgw1-az3

Configurazione CORS per HTTP APIs in API Gateway

La condivisione di risorse tra origini diverse (CORS) è una funzionalità di sicurezza del browser che limita HTTP le richieste avviate dagli script in esecuzione nel browser. Se non riesci ad accedere a API e ricevi un messaggio di errore che contiene`Cross-Origin Request Blocked`, potrebbe essere necessario abilitarlo. CORS Per ulteriori informazioni, consulta Cos'èCORS? .

CORSè in genere necessario per creare applicazioni Web che accedono APIs ospitate su un dominio o un'origine diversi. È possibile abilitare CORS l'autorizzazione alle richieste API provenienti da un'applicazione Web ospitata su un dominio diverso. Ad esempio, se il tuo API è ospitato su `https://{api_id}.execute-api.{region}.amazonaws.com/` e desideri chiamarti API da un'applicazione web ospitata su`example.com`, API devi supportareCORS.

Se CORS configuri per unAPI, API Gateway invia automaticamente una risposta alle OPTIONS richieste di preflight, anche se non esiste un OPTIONS percorso configurato per il tuoAPI. Per una CORS richiesta, API Gateway aggiunge le CORS intestazioni configurate alla risposta di un'integrazione.

> **Note**
>
> Se si configura CORS per unAPI, API Gateway ignora le CORS intestazioni restituite dall'integrazione con il backend.

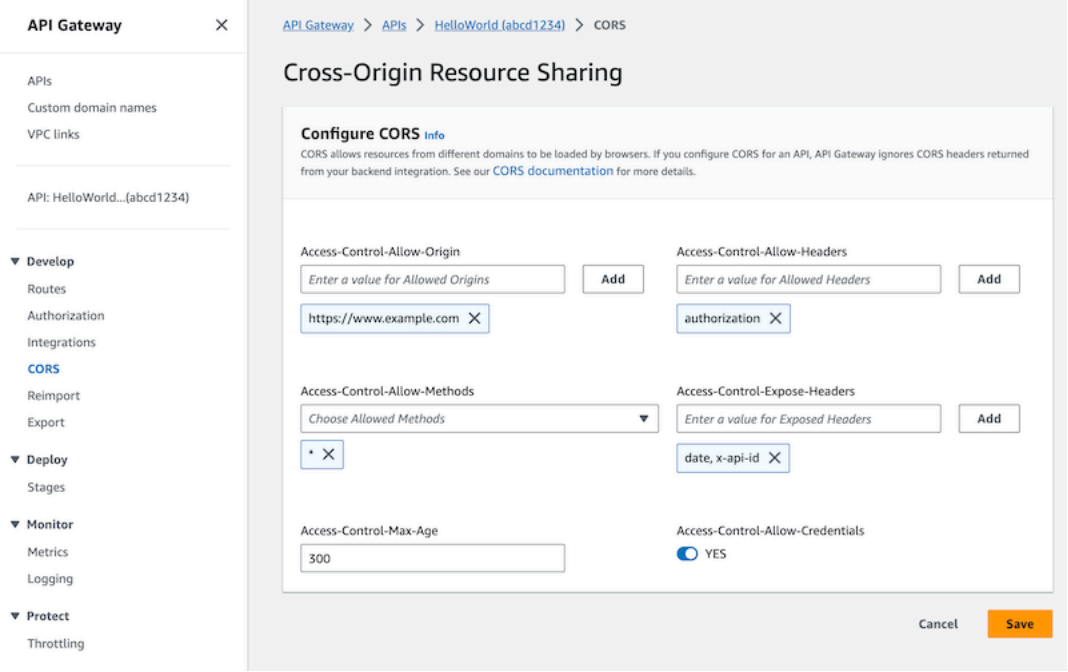
È possibile specificare i seguenti parametri in una configurazione. CORS Per aggiungere questi parametri utilizzando la HTTP API console API Gateway, scegli Aggiungi dopo aver inserito il valore.

CORSintestazioni	CORSproprietà di configurazione	Valori di esempio
Access-Control-Allow-Origin	allowOrigins	• `https://www.example.com` • `*` (consente tutte le origini) • `https://*` (consente qualsiasi origine che inizia con https://)

CORSintestazioni	CORSproprietà di configurazione	Valori di esempio
		• `http://*` (consente qualsiasi origine che inizia con http://)
Access-Control-Allow-Credentials	allowCredentials	true
Access-Control-Expose-Headers	exposeHeaders	Data x-api-id, *
Access-Control-Max-Age	maxAge	300
Access-Control-Allow-Methods	allowMethods	GET, POST, DELETE, *
Access-Control-Allow-Headers	allowHeaders	Authorization, *

Per restituire le CORS intestazioni, la richiesta deve contenere un'`origin`intestazione. Per il `OPTIONS` metodo, la richiesta deve contenere un'`origin`intestazione e un'intestazione. `Access-Control-Request-Method`

La tua CORS configurazione potrebbe essere simile alla seguente:



Configurazione CORS per un HTTP API con un **`$default`** percorso e un autorizzatore

È possibile abilitare CORS e configurare l'autorizzazione per qualsiasi percorso di un. HTTP API Quando si abilita CORS e si autorizza il `$default`percorso, ci sono alcune considerazioni speciali. Il percorso `$default` cattura le richieste per tutti i metodi e i percorsi che non sono stati definiti in modo esplicito, incluse le richieste `OPTIONS`. Per supportare `OPTIONS` le richieste non autorizzate, aggiungi al tuo `OPTIONS /{proxy+}` percorso un percorso API che non richieda l'autorizzazione e allega un'integrazione al percorso. Il percorso `OPTIONS /{proxy+}` ha priorità più alta rispetto al percorso `$default`. Di conseguenza, consente ai clienti di inviarti `OPTIONS` richieste API senza autorizzazione. Per ulteriori informazioni sulle priorità di instradamento, consultare Routing delle richieste API.

Configura CORS per e HTTP API utilizzando il AWS CLI

È possibile utilizzare il seguente comando update-api per abilitare CORS le richieste da. `https://www.example.com`

Example

```
aws apigatewayv2 update-api --api-id api-id --cors-configuration AllowOrigins="https://
www.example.com"
```

Per ulteriori informazioni, consulta CORSAmazon API Gateway Version 2 API Reference.

Trasforma API le richieste e le risposte per HTTP APIs in API Gateway

Puoi modificare le API richieste dei clienti prima che raggiungano le tue integrazioni di backend. Puoi anche modificare la risposta delle integrazioni prima che API Gateway restituisca la risposta ai client. Si utilizza la mappatura dei parametri per modificare le API richieste e le risposte per. HTTP APIs Per utilizzare la mappatura dei parametri, è necessario specificare i parametri di API richiesta o risposta da modificare e specificare come modificare tali parametri.

Trasformazione delle richieste API

È possibile utilizzare i parametri di richiesta per modificare le richieste prima che raggiungano le integrazioni back-end. È possibile modificare le intestazioni, le stringhe di query o il percorso della richiesta.

I parametri di richiesta sono una mappa chiave-valore. La chiave identificherà la posizione del parametro di richiesta da modificare e la modalità per modificarlo. Il valore specifica i nuovi dati per il parametro.

Nella tabella seguente sono illustrate le chiavi supportate.

Type	Sintassi
Intestazione	`append|overwrite|remove:header.` *`headername`*
Stringa di query	`append|overwrite|remove:querystring.` *`querystring-name`*
Percorso	`overwrite:path`

Nella tabella seguente sono illustrati i valori supportati che è possibile mappare ai parametri.

Type	Sintassi	Note
Valore intestazione	$request.header.*name* o ${request.header.*name*}	I nomi delle intestazioni non fanno distinzione tra maiuscole e minuscole. APIAd esempio, Gateway combina più valori di intestazione con virgole. `"header1": "value1,value2"` Alcune intestazioni sono riservate. Per ulteriori informazioni, consulta the section called “Intestazioni riservate”.
Valore della stringa di query	$request.querystring.*name* o ${request.querystring.*name*}	I nomi delle stringhe di query fanno distinzione tra maiuscole e minuscole. APIAd esempio, Gateway combina più valori con

Type	Sintassi	Note
		virgole. `"querystring1" "Value1,Value2"`
Corpo della richiesta	$request.body.*name* o ${request.body.*name*}	Un'espressione di JSON percorso. Le espressioni di discesa ricorsiva (`$request.body..name`) e di filtro (`?(expression)`) non sono supportate. **Note** Quando si specifica un JSON percorso, API Gateway tronca il corpo della richiesta a 100 KB e quindi applica l'espressione di selezione. Per inviare payload superiori a 100 KB, specificare `$request.body`.
Percorso della richiesta	$request.path o ${request.path}	Il percorso della richiesta, senza il nome della fase.
Parametro del percorso	$request.path.*name* o ${request.path.*name*}	Il valore di un parametro del percorso nella richiesta. Ad esempio, se l'instradamento è `/pets/{petId}`, è possibile mappare il parametro `petId` della richiesta con `$request.path.petId`.

Type	Sintassi	Note
Variabile di contesto	$contesto.*variableName* o $ {context.*variableName* }	Il valore di una variabile di contesto. **Note** Sono supportati solo i caratteri speciali `.` e `_`.
Variabile di fase	$stageVariables.*variableName* o $ {stageVariables.*variableName* }	Il valore di una variabile di fase.
Valore statico	*string*	Un valore costante.

> **Note**
> Per utilizzare più variabili in un'espressione di selezione, racchiudere la variabile tra parentesi. Ad esempio, `${request.path.name} ${request.path.id}`.

Trasformare le risposte API

Si utilizzano i parametri di risposta per trasformare la HTTP risposta da un'integrazione di backend prima di restituire la risposta ai client. È possibile modificare le intestazioni o il codice di stato di una risposta prima che API Gateway restituisca la risposta ai client.

Configurare i parametri di risposta per ogni codice di stato restituito dall'integrazione. I parametri di risposta sono una mappa chiave-valore. La chiave identificherà la posizione del parametro di richiesta da modificare e la modalità per modificarlo. Il valore specifica i nuovi dati per il parametro.

Nella tabella seguente sono illustrate le chiavi supportate.

Type	Sintassi
Intestazione	`append|overwrite|remove:header.` *`headername`*
Codice di stato	`overwrite:statuscode`

Nella tabella seguente sono illustrati i valori supportati che è possibile mappare ai parametri.

Type	Sintassi	Note
Valore intestazione	$response.header.*name* o ${response.header.*name*}	I nomi delle intestazioni non fanno distinzione tra maiuscole e minuscole. APIAd esempio, Gateway combina più valori di intestazione con virgole. `"header1": "value1,value2"` Alcune intestazioni sono riservate. Per ulteriori informazioni, consulta the section called “Intestazioni riservate”.
Corpo di risposta	$response.body.*name* o ${response.body.*name*}	Un'JSONespressione di percorso. Le espressioni di discesa ricorsiva (`$response.body..name`) e di filtro (`?(expression)`) non sono supportate. **Note** Quando si specifica un JSON percorso, API Gateway tronca il corpo della risposta a 100 KB e quindi

Type	Sintassi	Note
		applica l'espressione di selezione. Per inviare payload superiori a 100 KB, specificare `$response.body` .
Variabile di contesto	$contesto.*`variableName`* o $ {context.*`variableName`* }	Valore di una variabile di contesto supportata.
Variabile di fase	$stageVariables.*`variableName`* o $ {stageVariables.*`variableName`* }	Il valore di una variabile di fase.
Valore statico	*`string`*	Un valore costante.

> **Note**
>
> Per utilizzare più variabili in un'espressione di selezione, racchiudere la variabile tra parentesi. Ad esempio, `${request.path.name} ${request.path.id}`.

Intestazioni riservate

Le intestazioni seguenti sono riservate. Per queste intestazioni, non è possibile configurare le mappature di richiesta o di risposta.

- access-control-*
- apigw-*
- Autorizzazione
- Connessione
- Content-Encoding
- Content-Length
- Content-Location

- Forwarded
- Keep-Alive
- Origin
- Proxy-Authenticate
- Proxy-Authorization
- TE
- Trailers
- Transfer-Encoding
- Upgrade
- x-amz-*
- x-amzn-*
- X-Forwarded-For
- X-Forwarded-Host
- X-Forwarded-Proto
- Via

Esempi

I seguenti AWS CLI esempi configurano le mappature dei parametri. Per esempi di AWS CloudFormation modelli, vedere. GitHub

Aggiungere un'intestazione a una richiesta API

L'esempio seguente aggiunge un'intestazione denominata `header1` a una API richiesta prima che raggiunga l'integrazione del backend. APIGateway compila l'intestazione con l'ID della richiesta.

```
aws apigatewayv2 create-integration \
    --api-id abcdef123 \
    --integration-type HTTP_PROXY \
    --payload-format-version 1.0 \
    --integration-uri 'https://api.example.com' \
    --integration-method ANY \
    --request-parameters '{ "append:header.header1": "$context.requestId" }'
```

Rinominare un'intestazione di richiesta

Nell'esempio seguente, viene rinominata un'intestazione di richiesta da `header1` a `header2`.

```
aws apigatewayv2 create-integration \
    --api-id abcdef123 \
    --integration-type HTTP_PROXY \
    --payload-format-version 1.0 \
    --integration-uri 'https://api.example.com' \
    --integration-method ANY \
    --request-parameters '{ "append:header.header2": "$request.header.header1",
 "remove:header.header1": "'"}'
```

Modificare la risposta da un'integrazione

Nell'esempio seguente, vengono configurati i parametri di risposta per un'integrazione. Quando le integrazioni restituiscono un codice di stato 500, API Gateway modifica il codice di stato in 403 e aggiunge `header1` 1 alla risposta. Quando l'integrazione restituisce un codice di stato 404, API Gateway aggiunge un'`error`intestazione alla risposta.

```
aws apigatewayv2 create-integration \
    --api-id abcdef123 \
    --integration-type HTTP_PROXY \
    --payload-format-version 1.0 \
    --integration-uri 'https://api.example.com' \
    --integration-method ANY \
    --response-parameters '{"500" : {"append:header.header1": "$context.requestId",
 "overwrite:statuscode" : "403"}, "404" : {"append:header.error" :
 "$stageVariables.environmentId"}  }'
```

Rimuovere le mappature dei parametri configurati

Il comando di esempio seguente rimuove i parametri di richiesta configurati in precedenza per `append:header.header1`. Il comando rimuove anche i parametri di risposta configurati in precedenza per un codice di stato 200.

```
aws apigatewayv2 update-integration \
    --api-id abcdef123 \
    --integration-id hijk456 \
    --request-parameters '{"append:header.header1" : ""}' \
    --response-parameters '{"200" : {}}'
```

Usa le definizioni OpenAPI per le API HTTP in API Gateway

È possibile definire l'API HTTP utilizzando un file di definizione OpenAPI 3.0. Quindi è possibile importare la definizione in API Gateway per creare un'API. Per ulteriori informazioni sulle estensioni di API Gateway verso OpenAPI, consultare APIEstensioni aperte.

Importazione di un'API HTTP

È possibile creare un'API HTTP importando un file di definizione OpenAPI 3.0.

Per eseguire la migrazione da un'API REST a un'API HTTP, è possibile esportare l'API REST come file di definizione OpenAPI 3.0. Quindi importare la definizione dell'API come un'API HTTP. Per ulteriori informazioni sull'esportazione di un'API REST, consultare Esporta un file REST API da API Gateway.

> **Note**
>
> Le API HTTP supportano le stesse AWS variabili delle API REST. Per ulteriori informazioni, consulta AWS variabili per Open import API.

Informazioni di convalida dell'importazione

Quando si importa un'API, API Gateway fornisce tre categorie di informazioni di convalida.

Info

Una proprietà è valida in base alla specifica OpenAPI, ma tale proprietà non è supportata per le API HTTP.

Ad esempio, la seguente porzione di codice di OpenAPI 3.0 produce informazioni sull'importazione perché le API HTTP non supportano la convalida della richiesta. API Gateway ignora i campi `requestBody` e `schema`.

```
"paths": {
  "/": {
    "get": {
      "x-amazon-apigateway-integration": {
        "type": "AWS_PROXY",
        "httpMethod": "POST",
        "uri": "arn:aws:lambda:us-east-2:123456789012:function:HelloWorld",
        "payloadFormatVersion": "1.0"
```

```
      },
      "requestBody": {
        "content": {
          "application/json": {
            "schema": {
              "$ref": "#/components/schemas/Body"
            }
          }
        }
      }
    }
  }
  ...
},
"components": {
  "schemas": {
    "Body": {
      "type": "object",
      "properties": {
        "key": {
          "type": "string"
        }
      }
    }
    ...
  }
  ...
}
```

Attenzione

Una proprietà o una struttura non è valida in base alla specifica OpenAPI, ma non blocca la creazione di API. È possibile specificare se l'API Gateway deve ignorare questi avvisi e continuare a creare l'API o interrompere la creazione dell'API n caso di presenza di avvisi.

Il seguente documento OpenAPI 3.0 produce avvisi sull'importazione perché le API HTTP supportano solo le integrazioni proxy Lambda e proxy HTTP.

```
"x-amazon-apigateway-integration": {
  "type": "AWS",
  "httpMethod": "POST",
  "uri": "arn:aws:lambda:us-east-2:123456789012:function:HelloWorld",
  "payloadFormatVersion": "1.0"
```

```
}
```

Errore

La specifica OpenAPI non è valida o è malformata. API Gateway non è in grado di creare alcuna risorsa dal documento malformato. È necessario correggere gli errori e quindi riprovare.

La seguente definizione API produce errori durante l'importazione perché le API HTTP supportano solo la specifica OpenAPI 3.0.

```
{
  "swagger": "2.0.0",
  "info": {
    "title": "My API",
    "description": "An Example OpenAPI definition for Errors/Warnings/ImportInfo",
    "version": "1.0"
  }
  ...
}
```

Come altro esempio, mentre OpenAPI consente agli utenti di definire un'API con più requisiti di sicurezza associati a una particolare operazione, API Gateway non supporta questa operazione. Ogni operazione può avere solo un'autorizzazione IAM, un provider di autorizzazioni Lambda o un provider di autorizzazioni JWT. Il tentativo di modellare più requisiti di sicurezza genera un errore.

Importa un'API utilizzando il AWS CLI

Il seguente comando importa il file di definizione OpenAPI 3.0 `api-definition.json` come sotto forma di API HTTP.

Example

```
aws apigatewayv2 import-api --body file://api-definition.json
```

Example

È possibile importare la seguente definizione OpenAPI 3.0 di esempio per creare un'API HTTP.

```
{
  "openapi": "3.0.1",
  "info": {
```

```
    "title": "Example Pet Store",
    "description": "A Pet Store API.",
    "version": "1.0"
  },
  "paths": {
    "/pets": {
      "get": {
        "operationId": "GET HTTP",
        "parameters": [
          {
            "name": "type",
            "in": "query",
            "schema": {
              "type": "string"
            }
          },
          {
            "name": "page",
            "in": "query",
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "headers": {
              "Access-Control-Allow-Origin": {
                "schema": {
                  "type": "string"
                }
              }
            },
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Pets"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
```

```
          "type": "HTTP_PROXY",
          "httpMethod": "GET",
          "uri": "http://petstore.execute-api.us-west-1.amazonaws.com/petstore/pets",
          "payloadFormatVersion": 1.0
        }
      },
      "post": {
        "operationId": "Create Pet",
        "requestBody": {
          "content": {
            "application/json": {
              "schema": {
                "$ref": "#/components/schemas/NewPet"
              }
            }
          },
          "required": true
        },
        "responses": {
          "200": {
            "description": "200 response",
            "headers": {
              "Access-Control-Allow-Origin": {
                "schema": {
                  "type": "string"
                }
              }
            },
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/NewPetResponse"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "HTTP_PROXY",
          "httpMethod": "POST",
          "uri": "http://petstore.execute-api.us-west-1.amazonaws.com/petstore/pets",
          "payloadFormatVersion": 1.0
        }
      }
```

```
    },
    "/pets/{petId}": {
      "get": {
        "operationId": "Get Pet",
        "parameters": [
          {
            "name": "petId",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "headers": {
              "Access-Control-Allow-Origin": {
                "schema": {
                  "type": "string"
                }
              }
            },
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Pet"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "HTTP_PROXY",
          "httpMethod": "GET",
          "uri": "http://petstore.execute-api.us-west-1.amazonaws.com/petstore/pets/
{petId}",
          "payloadFormatVersion": 1.0
        }
      }
    }
  },
  "x-amazon-apigateway-cors": {
```

```
    "allowOrigins": [
      "*"
    ],
    "allowMethods": [
      "GET",
      "OPTIONS",
      "POST"
    ],
    "allowHeaders": [
      "x-amzm-header",
      "x-apigateway-header",
      "x-api-key",
      "authorization",
      "x-amz-date",
      "content-type"
    ]
  },
  "components": {
    "schemas": {
      "Pets": {
        "type": "array",
        "items": {
          "$ref": "#/components/schemas/Pet"
        }
      },
      "Empty": {
        "type": "object"
      },
      "NewPetResponse": {
        "type": "object",
        "properties": {
          "pet": {
            "$ref": "#/components/schemas/Pet"
          },
          "message": {
            "type": "string"
          }
        }
      },
      "Pet": {
        "type": "object",
        "properties": {
          "id": {
            "type": "string"
```

```
          },
          "type": {
            "type": "string"
          },
          "price": {
            "type": "number"
          }
        }
      },
      "NewPet": {
        "type": "object",
        "properties": {
          "type": {
            "$ref": "#/components/schemas/PetType"
          },
          "price": {
            "type": "number"
          }
        }
      },
      "PetType": {
        "type": "string",
        "enum": [
          "dog",
          "cat",
          "fish",
          "bird",
          "gecko"
        ]
      }
    }
  }
}
```

Esportazione di API HTTP da API Gateway

Dopo aver creato un'API HTTP, è possibile esportare una definizione OpenAPI 3.0 della propria API da API Gateway. È possibile scegliere una fase da esportare o esportare l'ultima configurazione dell'API. È inoltre possibile importare una definizione API esportata in API Gateway per creare un'altra API identica. Per ulteriori informazioni sull'importazione delle definizioni API, consultare Importazione di un'API HTTP.

Esportazione di una definizione OpenAPI 3.0 di uno stage utilizzando la CLI AWS

Il comando seguente esporta una definizione OpenAPI di una fase API denominata `prod` in un file YAML denominato `stage-definition.yaml`. Per impostazione predefinita il file di definizione esportato include le estensioni API Gateway.

```
aws apigatewayv2 export-api \
    --api-id api-id  \
    --output-type YAML  \
    --specification OAS30 \
    --stage-name prod \
    stage-definition.yaml
```

Esporta una definizione OpenAPI 3.0 delle ultime modifiche della tua API utilizzando la CLI AWS

Il comando seguente esporta una definizione OpenAPI di un'API HTTP in un file JSON denominato `latest-api-definition.json`. Poiché il comando non specifica una fase, API Gateway esporta la configurazione più recente dell'API, indipendentemente dal fatto che sia stata distribuita in una fase o meno. Il file di definizione esportato non include le estensioni API Gateway.

```
aws apigatewayv2 export-api \
    --api-id api-id  \
    --output-type JSON  \
    --specification OAS30 \
    --no-include-extensions \
    latest-api-definition.json
```

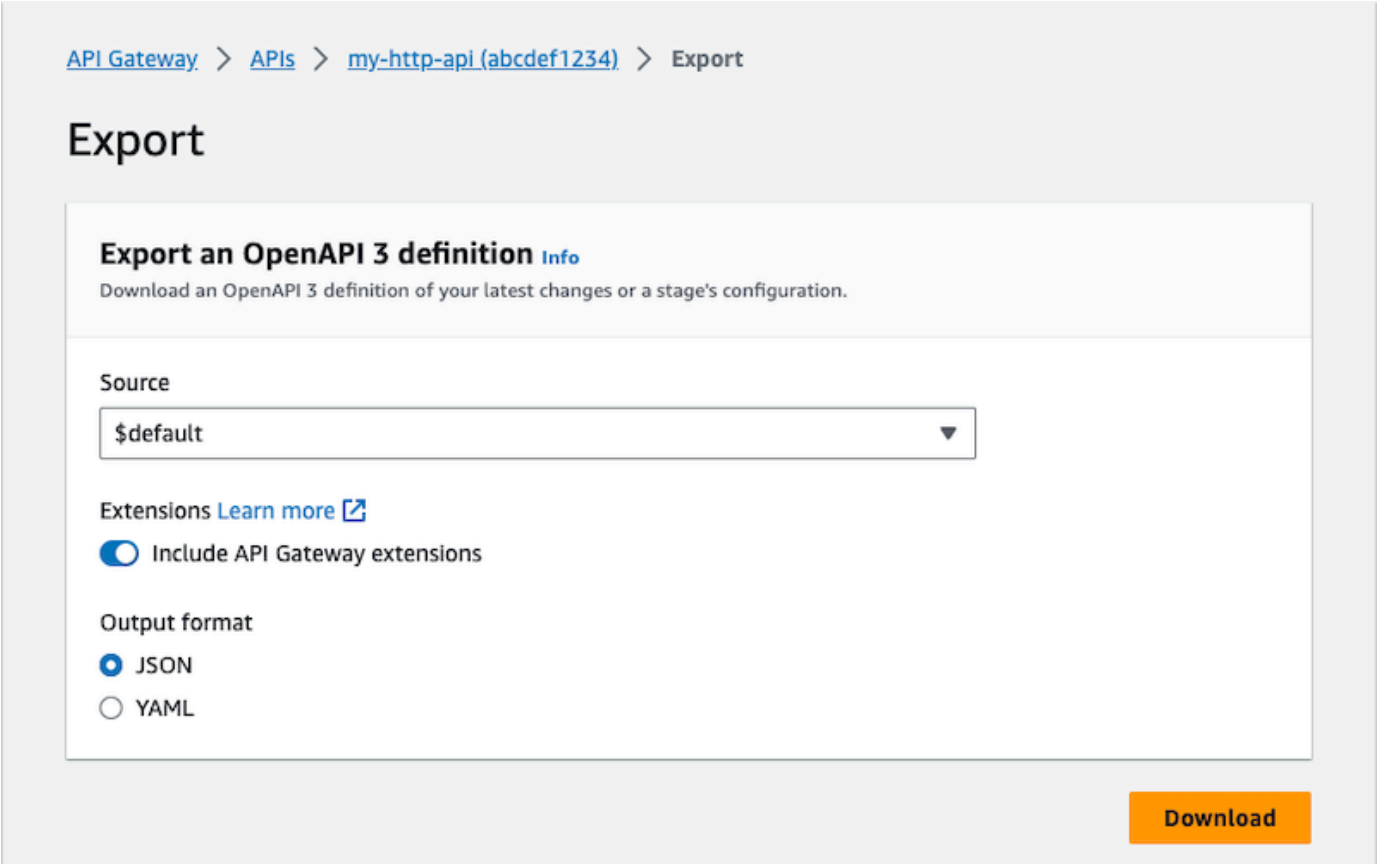
Per ulteriori informazioni, consulta ExportAPI nella Guida di riferimento per le API di Amazon API Gateway Versione 2.

Esportazione di una definizione OpenAPI 3.0 utilizzando la console Gateway API

La seguente procedura mostra come esportare una definizione OpenAPI di un'API HTTP.

Esportazione di una definizione OpenAPI 3.0 utilizzando la console Gateway API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere un'API HTTP.
3. Nel pannello di navigazione principale, in Sviluppa, scegli Esporta.
4. Seleziona una delle seguenti opzioni per esportare la tua API:



a. In Origine, seleziona un'origine per la definizione OpenAPI 3.0. Puoi scegliere una fase da esportare o esportare l'ultima configurazione dell'API.

b. Attiva Includi le estensioni API Gateway per includere le estensioni Gateway API.

c. In Formato di output, seleziona un formato di output.

5. Scegli Download (Scarica).

Pubblica API HTTP che i clienti possono richiamare

Puoi utilizzare fasi e nomi di dominio personalizzati per pubblicare l'API richiamabile dai client.

Una fase API è un riferimento logico a uno stato del ciclo di vita dell'API (ad esempio, `dev`, `prod`, `beta` o `v2`). Ogni fase è un riferimento con nome a una distribuzione dell'API e viene resa disponibile per le applicazioni client da chiamare. Puoi configurare integrazioni e impostazioni differenti per ogni fase di un'API.

Puoi utilizzare nomi di dominio personalizzati per fornire un URL più semplice e intuitivo all'API richiamabile dai client, rispetto all'URL predefini, `https://api-id.execute-api.region.amazonaws.com/stage`.

> **Note**
>
> Per aumentare la sicurezza delle API API Gateway, il dominio `execute-api.{region}.amazonaws.com`è registrato nella Public Suffix List (PSL). Per una maggiore sicurezza, consigliamo di utilizzare i cookie con un prefisso `__Host-` se hai bisogno di impostare cookie sensibili nel nome di dominio predefinito per le API API Gateway. Questa pratica ti aiuterà a difendere il tuo dominio dai tentativi CSRF (cross-site request forgery). Per ulteriori informazioni, consulta la pagina Impostazione cookie nella pagina Mozilla Developer Network.

Argomenti

- Fasi per HTTP APIs in API Gateway
- Politica di sicurezza per HTTP APIs in API Gateway
- Nomi di dominio personalizzati per HTTP APIs in API Gateway

Fasi per HTTP APIs in API Gateway

Una API fase è un riferimento logico a uno stato del ciclo di vita dell'utente API (ad esempio,, `dev` `prodbeta`, `ov2`). APIIe fasi sono identificate dal relativo API ID e dal nome della fase e sono incluse nel file utilizzato per richiamare ilURL. API Ogni fase è un riferimento denominato a una distribuzione di API ed è resa disponibile per essere richiamata dalle applicazioni client.

Puoi creare una `$default` fase che viene servita dalla base API della tua, `https://{api_id}.execute-api.{region}.amazonaws.com/` ad URL esempio. Lo usi URL per richiamare uno stageAPI.

Una distribuzione è un'istantanea della configurazioneAPI. Dopo averlo distribuito su una API fase, è disponibile per essere richiamato dai client. È necessario distribuire un file API affinché le modifiche abbiano effetto. Se abiliti le distribuzioni automatiche, le modifiche a an API vengono rilasciate automaticamente.

Utilizza le variabili di fase per HTTP APIs in Gateway API

Le variabili di fase sono coppie chiave-valore che è possibile definire per uno stadio di un. HTTP API Agiscono come variabili di ambiente e possono essere utilizzate nella configurazioneAPI.

Le variabili di fase non sono destinate ad essere utilizzate per i dati sensibili, come le credenziali. Per trasferire dati sensibili alle integrazioni, usa un AWS Lambda autorizzatore. È possibile passare dati sensibili alle integrazioni nell'output del provider di autorizzazioni Lambda. Per ulteriori informazioni, consulta the section called “Formato della risposta dell'autorizzazione”.

Esempio: utilizza una variabile di fase per personalizzare l'HTTPendpoint di integrazione

Ad esempio, è possibile definire una variabile di fase e quindi impostarne il valore come HTTP endpoint per un'integrazione HTTP proxy. Successivamente, puoi fare riferimento all'endpoint utilizzando il nome della variabile di fase associata. In questo modo, è possibile utilizzare la stessa API configurazione con un endpoint diverso in ogni fase. Allo stesso modo, potete utilizzare le variabili di fase per specificare una diversa integrazione di AWS Lambda funzioni per ogni fase del vostroAPI.

Per utilizzare una variabile stage per personalizzare l'endpoint di HTTP integrazione, dovete prima impostare il nome e il valore della variabile stage (ad esempio,`url`) con un valore di`example.com`. Quindi, configura un'integrazione HTTP proxy. Invece di inserire quelli dell'endpointURL, puoi dire a API Gateway di usare il valore della variabile stage,**`http://${stageVariables.url}`**. Questo valore indica a API Gateway di sostituire la variabile stage `${}` in fase di esecuzione, a seconda della fase del processo. API

È possibile fare riferimento alle variabili di fase in modo simile per specificare il nome o il ruolo di una funzione Lambda. AWS ARN

Quando si specifica un nome di funzione Lambda come valore della variabile di fase, è necessario configurare manualmente le autorizzazioni per la funzione Lambda. A tale scopo, è possibile utilizzare il seguente AWS CLI comando.

```
aws lambda add-permission --function-name arn:aws:lambda:XXXXXX:your-lambda-function-
name --source-arn arn:aws:execute-api:us-east-1:YOUR_ACCOUNT_ID:api_id/*/HTTP_METHOD/
resource --principal apigateway.amazonaws.com --statement-id apigateway-access --action
 lambda:InvokeFunction
```

APIRiferimento alle variabili di fase Gateway per HTTP APIs in API Gateway

È possibile utilizzare le variabili di fase API Gateway per HTTP APIs nei seguenti casi.

HTTPintegrazione URIs

È possibile utilizzare una variabile stage come parte di un'HTTPintegrazioneURI, come illustrato negli esempi seguenti.

- Una versione completa URI senza protocollo — `http://${stageVariables.<variable_name>}`
- Un dominio completo – `http://${stageVariables.<variable_name>}/resource/operation`
- Un sottodominio – `http://${stageVariables.<variable_name>}.example.com/resource/operation`
- Un percorso – `http://example.com/${stageVariables.<variable_name>}/bar`
- Una stringa di query – `http://example.com/foo?q=${stageVariables.<variable_name>}`

Funzioni Lambda

Puoi utilizzare una variabile di fase al posto di un nome di integrazione di funzione o alias Lambda, come illustrato negli esempi seguenti.

- `arn:aws:apigateway:<region>:lambda:path/2015-03-31/functions/arn:aws:lambda:<region>:<account_id>:function:${stageVariables.<function_variable_name>}/invocations`
- `arn:aws:apigateway:<region>:lambda:path/2015-03-31/functions/arn:aws:lambda:<region>:<account_id>:function:<function_name>:${stageVariables.<version_variable_name>}/invocations`

> **Note**
>
> Per utilizzare una variabile stage per una funzione Lambda, la funzione deve trovarsi nello stesso account di. API Le variabili di fase non supportano le funzioni Lambda tra più account.

AWS credenziali di integrazione

È possibile utilizzare una variabile di fase come parte di una credenziale AWS utente o di ruoloARN, come illustrato nell'esempio seguente.

- `arn:aws:iam::<account_id>:${stageVariables.<variable_name>}`

Politica di sicurezza per HTTP APIs in API Gateway

APIGateway applica una politica di sicurezza `TLS_1_2` per tutti gli HTTP API endpoint.

Una policy di sicurezza è una combinazione predefinita di TLS versioni minime e suite di crittografia offerte da Amazon Gateway. API Il TLS protocollo risolve problemi di sicurezza della rete come la manomissione e l'intercettazione tra client e server. Quando i client stabiliscono una TLS stretta di mano nei vostri confronti API tramite il dominio personalizzato, la politica di sicurezza applica la TLS versione e le opzioni della suite di crittografia che i clienti possono scegliere di utilizzare. Questa politica di sicurezza accetta il traffico TLS 1.2 e TLS 1.3 e rifiuta il traffico 1.0. TLS

TLSProtocolli e cifrari supportati per HTTP APIs

La tabella seguente descrive i TLS protocolli supportati per. HTTP APIs

TLSprotocolli	TLS_1_2 politica di sicurezza
TLSv1.3	Sì
TLSv12.	Sì

La tabella seguente descrive i codici TLS disponibili per la politica di sicurezza TLS 1_2 per. HTTP APIs

TLScifrari	TLS_1_2 politica di sicurezza
TLS- -128- - AES GCM SHA256	Sì
TLS- AES -256- - GCM SHA384	Sì

TLScifrari	TLS_1_2 politica di sicurezza
TLS- CHACHA2 0- POLY13 05- SHA256	Sì
ECDHE-ECDSA-AES128-GCM-SHA256	Sì
ECDHE-RSA-AES128-GCM-SHA256	Sì
ECDHE-ECDSA-AES128-SHA256	Sì
ECDHE-RSA-AES128-SHA256	Sì
ECDHE-ECDSA-AES256-GCM-SHA384	Sì
ECDHE-RSA-AES256-GCM-SHA384	Sì
ECDHE-ECDSA-AES256-SHA384	Sì
ECDHE-RSA-AES256-SHA384	Sì
AES128-GCM-SHA256	Sì
AES128-SHA256	Sì
AES256-GCM-SHA384	Sì

TLScifrari	TLS_1_2 politica di sicurezza
AES256-SHA256	Sì

Nomi aperti SSL e RFC cifrati

Open SSL e IETF RFC 5246 utilizzano nomi diversi per gli stessi cifrari. Per un elenco dei nomi dei cifrari, vedere. the section called “Nomi aperti SSL e cifrati RFC”

Informazioni su e REST APIs WebSocket APIs

Per ulteriori informazioni su REST APIs e WebSocket APIs, vedere the section called “Scegli una politica di sicurezza” ethe section called “Politica di sicurezza per WebSocket APIs in API Gateway”.

Nomi di dominio personalizzati per HTTP APIs in API Gateway

I nomi di dominio personalizzati sono più semplici e intuitivi URLs che puoi fornire ai tuoi API utenti.

Dopo aver distribuito il tuoAPI, tu (e i tuoi clienti) potete richiamarlo API utilizzando la base predefinita URL del seguente formato:

```
https://api-id.execute-api.region.amazonaws.com/stage
```

dove *`api-id`* è generato da API Gateway, *`region`* è la AWS regione, e *`stage`* è specificata dall'utente durante la API distribuzione di.

La parte hostname diURL, si *`api-id`*`.execute-api.`*`region`*`.amazonaws.com` riferisce a un API endpoint. Il nome dell'APIendpoint predefinito viene generato casualmente, è difficile da richiamare e non è intuitivo.

Con i nomi di dominio personalizzati, puoi configurare il tuo hostname e scegliere un percorso API di base (ad esempio,`myservice`) per mappare l'alternativa al tuo. URL API Ad esempio, una API base più intuitiva URL può diventare:

```
https://api.example.com/myservice
```

Considerazioni

Le seguenti considerazioni potrebbero influire sull'utilizzo di un nome di dominio personalizzato.

- Un nome di dominio personalizzato regionale può essere associato a REST APIs e HTTPAPIs. È possibile utilizzare la versione 2 del API Gateway APIs per creare e gestire nomi di dominio personalizzati regionali per RESTAPIs.
- Per la TLS versione minima, è supportata solo la versione TLS 1.2.
- È necessario creare o aggiornare il record di risorse del DNS provider da mappare all'APIendpoint. Senza tale mappatura, API le richieste associate al nome di dominio personalizzato non possono raggiungere API Gateway.
- È possibile supportare un numero quasi infinito di nomi di dominio senza superare la quota predefinita utilizzando un certificato wildcard. Per ulteriori informazioni, consulta the section called “Nomi di dominio personalizzati con caratteri jolly”.

Prerequisiti

Di seguito sono riportati i prerequisiti per la creazione di un nome di dominio personalizzato.

Registrare un nome di dominio

È necessario disporre di un nome di dominio Internet registrato per configurare nomi di dominio personalizzati per il proprioAPIs. Puoi registrare il tuo nome di dominio Internet utilizzando Amazon Route 53 o utilizzando un registrar di domini di terze parti a tua scelta. Il tuo nome di dominio personalizzato può essere il nome di un sottodominio o il dominio principale (noto anche come «apice di zona») di un dominio Internet registrato.

Il nome di dominio deve seguire la specifica RFC1035 e può avere un massimo di 63 ottetti per etichetta e 255 ottetti in totale.

Certificati per nomi di dominio personalizzati

Prima di configurare un nome di dominio personalizzato per unAPI, è necessario disporre di un certificato/pronto all'uso. SSL TLS ACM Se non ACM è disponibile nella AWS regione in cui si sta creando il nome di dominio personalizzato, è necessario importare un certificato in API Gateway in quella regione.

Per importare un TLS certificatoSSL/, è necessario fornire il corpo del TLS certificato PEM SSL / formattato, la relativa chiave privata e la catena di certificati per il nome di dominio personalizzato.

Ogni certificato memorizzato in ACM è identificato dal relativo. ARN Con i certificati emessi daACM, non devi preoccuparti di esporre dettagli sensibili del certificato, come la chiave privata. Per utilizzare un certificato AWS gestito per un nome di dominio, è sufficiente fare riferimento al relativoARN.

Se l'applicazione utilizza il blocco dei certificati, a volte noto come SSL pinning, per bloccare un ACM certificato, l'applicazione potrebbe non essere in grado di connettersi al dominio dopo il AWS rinnovo del certificato. Per ulteriori informazioni, consulta Probemi nel blocco dei certificati nella Guida per l'utente di AWS Certificate Manager .

Nomi di dominio personalizzati con caratteri jolly

Con i nomi di dominio personalizzati con caratteri jolly, è possibile supportare un numero quasi infinito di nomi di dominio senza superare la quota predefinita. Ad esempio, è possibile dare a ciascuno dei clienti il proprio nome di dominio, `customername.api.example.com`.

Per creare un nome di dominio personalizzato con caratteri jolly, specificare un carattere jolly (`*`) come primo sottodominio di un dominio personalizzato che rappresenta tutti i possibili sottodomini di un dominio radice.

Ad esempio, il nome di dominio personalizzato con caratteri jolly `*.example.com` genera sottodomini quali `a.example.com`, `b.example.com` e `c.example.com`, indirizzati tutti allo stesso dominio.

I nomi di dominio personalizzati Wildcard supportano configurazioni diverse dai nomi di dominio personalizzati standard di API Gateway. Ad esempio, in un singolo AWS account, è possibile configurare `*.example.com` e `a.example.com` comportarsi in modo diverso.

Per creare un nome di dominio personalizzato con caratteri wildcard, devi fornire un certificato emesso da ACM che sia stato convalidato utilizzando il metodo di convalida DNS o tramite e-mail.

> **Note**
>
> Non puoi creare un nome di dominio personalizzato con caratteri jolly se un altro AWS account ha creato un nome di dominio personalizzato che è in conflitto con il nome di dominio personalizzato con caratteri jolly. Ad esempio, se l'account A ha creato `a.example.com`, l'account B non può creare il nome di dominio personalizzato con caratteri jolly `*.example.com`.
> Se l'account A e l'account B condividono un proprietario, è possibile contattare il Centro assistenza AWS per richiedere un'eccezione.

Passaggi successivi per i nomi di dominio personalizzati

Per configurare un nome di dominio personalizzato per un HTTPAPI, utilizza la documentazione fornita REST API nella sezione della API Gateway Developer Guide.

Innanzitutto, specifica un certificato per il tuo nome di dominio personalizzato. Per ulteriori informazioni, consulta the section called “Prepara i certificati in AWS Certificate Manager”. Successivamente, crei un nome di dominio personalizzato regionale. Per ulteriori informazioni, consulta the section called “Configura un nome di dominio personalizzato regionale”.

Mappa le fasi dell'API su un nome di dominio personalizzato per le API HTTP

È possibile utilizzare le mappature API per connettere le fasi API a un nome di dominio personalizzato. Dopo aver creato un nome di dominio e aver configurato i record DNS, è possibile utilizzare le mappature API per inviare il traffico alle API tramite il nome di dominio personalizzato.

Una mappatura API specifica un'API, una fase e, facoltativamente, un percorso da utilizzare per la mappatura. Ad esempio, è possibile mappare la fase `production` di un'API su `https://api.example.com/orders`.

È possibile mappare le fasi HTTP e API REST allo stesso nome di dominio personalizzato.

Prima di creare una mappatura API, è necessario disporre di un'API, di una fase e di un nome di dominio personalizzato. Per ulteriori informazioni sulla creazione di un nome di dominio personalizzato, consulta the section called “Configura un nome di dominio personalizzato regionale”.

Routing delle richieste API

È possibile configurare le mappature API con più livelli, ad esempio `orders/v1/items` e `orders/v2/items`.

Per mappature API con più livelli, API Gateway instrada le richieste alla mappatura API che ha il percorso corrispondente più lungo. API Gateway considera solo i percorsi configurati per le mappature API, e non i percorsi API, per selezionare l'API da richiamare. Se nessun percorso corrisponde alla richiesta, API Gateway invia la richiesta all'API mappata al percorso vuoto `(none)`.

Per i nomi di dominio personalizzati che usano mappature API con più livelli, API Gateway instrada le richieste alla mappatura API che ha il percorso corrispondente più lungo.

Ad esempio, se si considera un nome di dominio personalizzato `https://api.example.com` con le seguenti mappature API:

1. `(none)` mappato all'API 1.
2. `orders` mappato all'API 2.
3. `orders/v1/items` mappato all'API 3.
4. `orders/v2/items` mappato all'API 4.
5. `orders/v2/items/categories` mappato all'API 5.

Richiesta	API selezionata	Spiegazione
`https://api.example.com/orders`	`API 2`	La richiesta corrisponde esattamente a questa mappatura API.
`https://api.example.com/orders/v1/items`	`API 3`	La richiesta corrisponde esattamente a questa mappatura API.
`https://api.example.com/orders/v2/items`	`API 4`	La richiesta corrisponde esattamente a questa mappatura API.
`https://api.example.com/orders/v1/items/123`	`API 3`	API Gateway sceglie la mappatura con il percorso corrispondente più lungo. `123` alla fine della richiesta non influisce sulla selezione.
`https://api.example.com/orders/v2/items/categories/5`	`API 5`	API Gateway sceglie la mappatura con il percorso corrispondente più lungo.
`https://api.example.com/customers`	`API 1`	API Gateway utilizza la mappatura vuota come catch-all.
`https://api.example.com/ordersandmore`	`API 2`	API Gateway sceglie la mappatura con il prefisso corrispondente più lungo.

Richiesta	API selezionata	Spiegazione
		Per un nome di dominio personalizzato configurato con mappature a livello singolo, ad esempio solo `https://api.example.com/orders` e `https://api.example.com/` , API Gateway sceglie `API 1`, poiché non esiste un percorso corrispondente con `ordersandmore` .

Restrizioni

- In una mappatura API, il nome di dominio personalizzato e le API mappate devono trovarsi nello stesso account. AWS
- Le mappature API devono contenere solo lettere, numeri e i seguenti caratteri: `$-_.+!*'()/`.
- La lunghezza massima per il percorso in una mappatura API è di 300 caratteri.
- È possibile disporre di 200 mappature API con più livelli per ogni nome di dominio.
- È possibile mappare le API HTTP a un nome di dominio personalizzato regionale solo con la policy di sicurezza TLS 1.2.
- Non WebSocket è possibile mappare le API allo stesso nome di dominio personalizzato di un'API HTTP o di un'API REST.

Creare una mappatura API

Per creare una mappatura API, innanzitutto è necessario creare un nome di dominio personalizzato, un'API e una fase. Per informazioni sulla creazione di un nome di dominio personalizzato, consulta the section called “Configura un nome di dominio personalizzato regionale”.

Per esempio i AWS Serverless Application Model modelli che creano tutte le risorse, vedi Sessions With SAM on GitHub.

AWS Management Console

Per creare una mappatura API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere Nomi di dominio personalizzati.
3. Selezionare un nome di dominio personalizzato già creato.
4. Scegliere API mappings (mappature API).
5. Scegliere Configure API mappings (Configura mappature API).
6. Scegliere Add new mapping (Aggiungi nuova mappatura).
7. Immettere un'API, uno Stage (Fase)e, facoltativamente, un Path (Percorso).
8. Selezionare Salva.

AWS CLI

Il AWS CLI comando seguente crea una mappatura delle API. In questo esempio, API Gateway invia le richieste `api.example.com/v1/orders` all'API e alla fase specificate.

```
aws apigatewayv2 create-api-mapping \
    --domain-name api.example.com \
    --api-mapping-key v1/orders \
    --api-id a1b2c3d4 \
    --stage test
```

AWS CloudFormation

L' AWS CloudFormation esempio seguente crea una mappatura delle API.

```
MyApiMapping:
  Type: 'AWS::ApiGatewayV2::ApiMapping'
  Properties:
    DomainName: api.example.com
    ApiMappingKey: 'orders/v2/items'
    ApiId: !Ref MyApi
    Stage: !Ref MyStage
```

Disabilita l'endpoint predefinito per le API HTTP

Per impostazione predefinita, i client possono richiamare l'API utilizzando l'endpoint `execute-api` generato da API Gateway per l'API. Per garantire che i client possano accedere all'API solo utilizzando un nome di dominio personalizzato con l'autenticazione TLS reciproca, disattivare l'endpoint `execute-api` predefinito. Quando si disattiva l'endpoint predefinito, questa operazione influisce su tutte le fasi di un'API.

La procedura seguente mostra come disabilitare l'endpoint predefinito per un'API HTTP.

AWS Management Console

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere un'API HTTP.
3. Scegli l'ID dell'API per aprire la pagina dei dettagli dell'API.
4. Nei dettagli dell'API, scegli Modifica.
5. Per Endpoint predefinito, seleziona Disabilita.
6. Selezionare Salva.

   Se attivi le distribuzioni automatiche per la tua fase, non è necessario ridistribuire l'API per rendere effettiva la modifica. Altrimenti, devi ridistribuire l'API.
7. (Facoltativo) Scegli Deploy, quindi ridistribuisci l'API o crea una nuova fase per rendere effettiva la modifica.

AWS CLI

Il update-domain-namecomando seguente disabilita l'endpoint predefinito per un'API HTTP.

```
aws apigatewayv2 update-api \
    --api-id abcdef123 \
    --disable-execute-api-endpoint
```

Dopo aver disattivato l'endpoint predefinito, è necessario distribuire l'API per rendere effettiva la modifica, a meno che non siano abilitate le distribuzioni automatiche.

Il seguente comando create-deployment crea una distribuzione.

```
aws apigatewayv2 create-deployment \
```

```
    --api-id abcdef123 \
    --stage-name dev
```

Proteggi le tue API HTTP in API Gateway

API Gateway fornisce diversi modi per proteggere l'API da determinate minacce, ad esempio utenti malintenzionati o picchi di traffico. È possibile proteggere l'API usando strategie come l'impostazione delle destinazioni di limitazione e l'abilitazione dell'autenticazione TLS reciproca. In questa sezione viene descritto come abilitare queste funzionalità utilizzando API Gateway.

Argomenti

- Limita le richieste alle tue API HTTP per una migliore velocità di trasmissione in API Gateway
- Come attivare l'TLSautenticazione reciproca per il tuo HTTP APIs in API Gateway

Limita le richieste alle tue API HTTP per una migliore velocità di trasmissione in API Gateway

È possibile configurare la limitazione della larghezza di banda della rete per le API per proteggerle da un numero eccessivo di richieste. Le limitazioni della larghezza di banda della rete sono applicate sulla base del massimo sforzo e dovrebbero essere considerate come obiettivi e non limiti massimi garantiti delle richieste.

API Gateway limita la larghezza di banda della rete delle richieste nell'API mediante l'algoritmo di token bucket, dove un token rappresenta una richiesta. Nello specifico, API Gateway esamina il tasso e i picchi di invio di richieste per tutte le API dell'account, per regione. Nell'algoritmo di bucket token, un picco può consentire il superamento predefinito di tali limiti, ma anche altri fattori possono causare il superamento dei limiti in alcuni casi.

Quando le richieste inviate superano i limiti impostati per il tasso a regime e per i limiti di picco, API Gateway inizia a limitare le richieste. I clienti potrebbero ricevere risposte agli errori `429 Too Many Requests` a questo punto. Quando rileva queste eccezioni, il client può reinviare le richieste non riuscite in modo da limitare la velocità.

In qualità di sviluppatore dell'API, puoi impostare i limiti per le singole fasi o le singole route dell'API per migliorare le prestazioni generali in tutte le API dell'account.

Limitazione a livello di account per regione

Per impostazione predefinita, API Gateway limita il numero di richieste con stato costante al secondo (RPS) per tutte le API all'interno di un account AWS , per Regione. Limita anche i picchi, ovvero la dimensione massima del bucket, per tutte le API di un account AWS , per regione. In API Gateway, il limite dei picchi rappresenta il numero massimo di invii di richieste che API Gateway può gestire prima di restituire risposte di errore `429 Too Many Requests`. Per ulteriori informazioni sulla limitazione delle quote, vedere Quote e note importanti.

I limiti per account vengono applicati a tutte le API di un account in una regione specificata. Il limite del tasso a livello di account può essere aumentato - sono possibili limiti più elevati con API con timeout più brevi e payload più piccoli. Per richiedere un aumento dei limiti di limitazione a livello di account per Regione, contatta il Centro assistenza AWS. Per ulteriori informazioni, consulta Quote e note importanti. Tieni presente che questi limiti non possono essere superiori ai limiti di AWS limitazione.

Throttling a livello di instradamento

Puoi impostare il throttling a livello di route per sostituire i limiti di throttling delle richieste a livello di account per una fase specifica o per singoli route nell'API. I limiti della limitazione (della larghezza di banda della rete) della route di default non possono superare i limiti di velocità a livello di account.

È possibile configurare la limitazione a livello di percorso utilizzando AWS CLI. Il comando seguente configura la limitazione personalizzata per lo stadio e la route specificati di un'API.

```
aws apigatewayv2 update-stage \
    --api-id a1b2c3d4 \
    --stage-name dev \
    --route-settings '{"GET /pets":
{"ThrottlingBurstLimit":100,"ThrottlingRateLimit":2000}}'
```

Come attivare l'TLSautenticazione reciproca per il tuo HTTP APIs in API Gateway

TLSL'autenticazione reciproca richiede l'autenticazione bidirezionale tra il client e il server. Con MutualTLS, i client devono presentare i certificati X.509 per verificare la propria identità e accedere al API Mutual TLS è un requisito comune per l'Internet of Things (IoT) e business-to-business le applicazioni.

È possibile utilizzare Mutual TLS insieme ad altre operazioni di autorizzazione e autenticazione supportate da API Gateway. APIGateway inoltra i certificati forniti dai client agli autorizzatori Lambda e alle integrazioni di backend.

> **Important**
>
> Per impostazione predefinita, i client possono richiamarti API utilizzando l'`execute-api`endpoint che Gateway genera per te. API API Per garantire che i clienti possano accedere al tuo API solo utilizzando un nome di dominio personalizzato con mutuoTLS, disabilita l'endpoint predefinito`execute-api`. Per ulteriori informazioni, consulta the section called “Disabilitazione dell'endpoint predefinito”.

Prerequisiti per la mutua TLS

Per configurare mutuo TLS è necessario:

- Un nome di dominio personalizzato
- Almeno un certificato configurato AWS Certificate Manager per il tuo nome di dominio personalizzato
- Un truststore configurato e caricato in Amazon S3

Nomi di dominio personalizzati

Per abilitare TLS il mutuo per a HTTPAPI, devi configurare un nome di dominio personalizzato per il tuoAPI. È possibile abilitare Mutual TLS per un nome di dominio personalizzato e quindi fornire il nome di dominio personalizzato ai client. Per accedere e utilizzare un nome API di dominio personalizzato con TLS attivazione reciproca, i client devono presentare certificati attendibili nelle API richieste. Puoi trovare ulteriori informazioni all'indirizzo the section called “Nomi di dominio personalizzati”.

Utilizzo dei certificati AWS Certificate Manager emessi

È possibile richiedere un certificato pubblicamente attendibile direttamente da ACM o importare certificati pubblici o autofirmati. Per configurare un certificato inACM, vai a ACM. Se desideri importare un certificato, continua a leggere la sezione seguente.

Utilizzo di un AWS Private Certificate Authority certificato o importato

Per utilizzare un certificato importato in ACM o da AWS Private Certificate Authority mutuoTLS, API Gateway necessita di un certificato `ownershipVerificationCertificate` emesso daACM. Questo certificato di proprietà viene utilizzato solo per verificare che disponi delle autorizzazioni per l'utilizzo del nome di dominio. Non viene utilizzato per la TLS stretta di mano. Se non ne hai già uno`ownershipVerificationCertificate`, vai https://console.aws.amazon.com/acm/a configurarne uno.

Dovrai mantenere la validità di questo certificato per tutta la durata del nome di dominio. Se un certificato scade e il rinnovo automatico non riesce, tutti gli aggiornamenti al nome di dominio verranno bloccati. Dovrai aggiornare il `ownershipVerificationCertificateArn` con un `ownershipVerificationCertificate` valido prima di poter apportare altre modifiche. `ownershipVerificationCertificate`Non può essere utilizzato come certificato server per un altro TLS dominio comune in API Gateway. Se un certificato viene reimportato direttamente inACM, l'emittente deve rimanere lo stesso.

Configurazione del truststore

I truststore sono file di testo con un'estensione `.pem`. Sono elenchi attendibili di certificati provenienti da autorità di certificazione. Per utilizzare mutuoTLS, crea un truststore di certificati X.509 di cui ti fidi per accedere ai tuoi. API

Devi includere nel truststore l'intera catena di attendibilità, a partire dal certificato della CA emittente, fino al certificato emesso da una CA radice. APIGateway accetta certificati client emessi da qualsiasi CA presente nella catena di fiducia. I certificati possono provenire da autorità di certificazione pubbliche o private. I certificati possono avere una lunghezza massima della catena di quattro. È inoltre possibile fornire certificati autofirmati. Nel truststore sono supportati i seguenti algoritmi di hashing:

- SHA-256 o superiore
- RSA-2048 o superiore
- ECDSA-256 o superiore

APIGateway convalida una serie di proprietà del certificato. È possibile utilizzare gli autorizzatori Lambda per eseguire controlli aggiuntivi quando un client richiama un certificatoAPI, incluso verificare se un certificato è stato revocato. APIGateway convalida le seguenti proprietà:

Validation	Descrizione
Sintassi X.509	Il certificato deve soddisfare i requisiti di sintassi X.509.
Integrità	Il contenuto del certificato non deve essere stato modificato rispetto a quello firmato dall'autorità di certificazione del truststore.
Validity	Il periodo di validità del certificato deve essere attuale.
Concatenamento di nomi/concatenamento di chiavi	I nomi e gli oggetti dei certificati devono formare una catena ininterrotta. I certificati possono avere una lunghezza massima della catena di quattro.

Carica il truststore in un bucket Amazon S3 in un singolo file

Example certificates.pem

```
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
...
```

Il AWS CLI comando seguente viene caricato nel tuo `certificates.pem` bucket Amazon S3.

```
aws s3 cp certificates.pem s3://bucket-name
```

Configurazione reciproca TLS per un nome di dominio personalizzato

Per configurare mutuo TLS per a HTTPAPI, è necessario utilizzare un nome di dominio personalizzato regionale per il proprioAPI, con una TLS versione minima di 1.2. Per ulteriori informazioni sulla creazione e la configurazione di un nome di dominio personalizzato, consulta the section called "Configura un nome di dominio personalizzato regionale".

> **Note**
>
> Mutual TLS non è supportato per uso privatoAPIs.

Dopo aver caricato il tuo truststore su Amazon S3, puoi configurare il tuo nome di dominio personalizzato per utilizzare mutuo. TLS Incolla quanto segue (barre incluse) in un terminale:

```
aws apigatewayv2 create-domain-name \
    --domain-name api.example.com \
    --domain-name-configurations CertificateArn=arn:aws:acm:us-
west-2:123456789012:certificate/123456789012-1234-1234-1234-12345678 \
    --mutual-tls-authentication TruststoreUri=s3://bucket-name/key-name
```

Dopo aver creato il nome di dominio, devi configurare i DNS record e le mappature dei percorsi di base per le operazioni. API Per ulteriori informazioni, consulta Configura un nome di dominio personalizzato regionale in API Gateway.

Invoca un utente API utilizzando un nome di dominio personalizzato che richiede mutuo TLS

Per richiamare un account API con mutuo TLS abilitato, i client devono presentare un certificato attendibile nella API richiesta. Quando un client tenta di richiamare il tuoAPI, API Gateway cerca l'emittente del certificato client nel tuo truststore. APIAffinché Gateway possa procedere con la richiesta, l'emittente del certificato e l'intera catena di fiducia fino al certificato CA principale devono trovarsi nel tuo truststore.

Il seguente comando `curl` di esempio seguente invia una richiesta `api.example.com,` che include `my-cert.pem` nella richiesta. `my-key.key` è la chiave privata del certificato.

```
curl -v --key ./my-key.key --cert ./my-cert.pem api.example.com
```

Il tuo API viene invocato solo se il truststore considera attendibile il certificato. Le seguenti condizioni faranno sì che API Gateway fallisca l'TLShandshake e neghi la richiesta con un codice di stato. `403` Se il tuo certificato:

- non è attendibile
- è scaduto
- non utilizza un algoritmo supportato

> **Note**
>
> APIGateway non verifica se un certificato è stato revocato.

Aggiornamento del truststore

Per aggiornare i certificati nel truststore, carica un nuovo bundle di certificati in Amazon S3. Potrai quindi aggiornare il nome di dominio personalizzato per utilizzare il certificato aggiornato.

Usa il Controllo delle versioni di Amazon S3 per mantenere più versioni del truststore. Quando aggiorni il tuo nome di dominio personalizzato per utilizzare una nuova versione del truststore, API Gateway restituisce avvisi se i certificati non sono validi.

APIGateway emette avvisi sui certificati solo quando aggiorni il nome di dominio. APIGateway non ti avvisa se un certificato precedentemente caricato scade.

Il AWS CLI comando seguente aggiorna un nome di dominio personalizzato per utilizzare una nuova versione di truststore.

```
aws apigatewayv2 update-domain-name \
    --domain-name api.example.com \
    --domain-name-configurations CertificateArn=arn:aws:acm:us-
west-2:123456789012:certificate/123456789012-1234-1234-1234-12345678 \
    --mutual-tls-authentication TruststoreVersion='abcdef123'
```

Disabilita mutuo TLS

Per disabilitare TLS il mutuo per un nome di dominio personalizzato, rimuovi il truststore dal tuo nome di dominio personalizzato, come mostrato nel comando seguente.

```
aws apigatewayv2 update-domain-name \
    --domain-name api.example.com \
    --domain-name-configurations CertificateArn=arn:aws:acm:us-
west-2:123456789012:certificate/123456789012-1234-1234-1234-12345678 \
    --mutual-tls-authentication TruststoreUri=''
```

Risolvi i problemi relativi a Mutual per TLS HTTP API

Di seguito vengono forniti consigli per la risoluzione di errori e problemi che potrebbero verificarsi quando si attiva Mutual. TLS

Risoluzione dei problemi relativi agli avvisi dei certificati

Quando si crea un nome di dominio personalizzato con mutuoTLS, API Gateway restituisce avvisi se i certificati nel truststore non sono validi. Questo può verificarsi anche quando aggiorni un nome di dominio personalizzato all'utilizzo di un nuovo truststore. Gli avvisi indicano il problema con il certificato e l'oggetto del certificato che ha generato l'avviso. Mutual TLS è ancora abilitato per teAPI, ma alcuni client potrebbero non essere in grado di accedervi. API

Per identificare il certificato che ha generato l'avviso, devi decodificare i certificati nel truststore. Puoi utilizzare strumenti come `openssl` per decodificare i certificati e identificarne gli oggetti.

Il comando seguente visualizza il contenuto di un certificato, incluso l'oggetto:

```
openssl x509 -in certificate.crt -text -noout
```

Aggiorna o rimuovi i certificati che hanno prodotto gli avvisi, quindi carica un nuovo truststore su Amazon S3. Dopo aver caricato il nuovo truststore, aggiorna il nome di dominio personalizzato per utilizzarlo.

Risoluzione dei problemi relativi ai conflitti dei nomi di dominio

L'errore `"The certificate subject <certSubject> conflicts with an existing certificate from a different issuer."` indica che più autorità di certificazione hanno emesso un certificato per questo dominio. Per ogni soggetto del certificato, può esserci un solo emittente in API Gateway per i TLS domini reciproci. Dovrai ottenere tutti i certificati per tale oggetto tramite un unico emittente. Se il problema riguarda un certificato di cui non hai il controllo ma puoi provare la proprietà del nome di dominio, contatta AWS Support per aprire un ticket.

Risoluzione dei problemi relativi ai messaggi di stato dei nomi di dominio

`PENDING_CERTIFICATE_REIMPORT`: Ciò significa che hai reimportato un certificato ACM e la convalida non è riuscita perché il nuovo certificato ha un SAN (nome alternativo del soggetto) che non è coperto dal `ownershipVerificationCertificate` o l'oggetto o SANs nel certificato non copre il nome di dominio. Potrebbe esserci una configurazione errata o è stato importato un certificato non valido. È necessario reimportare un certificato valido in. ACM Per ulteriori informazioni sulla convalida, consulta Convalida della proprietà del dominio.

`PENDING_OWNERSHIP_VERIFICATION`: Ciò significa che il certificato precedentemente verificato è scaduto e non ACM è stato possibile rinnovarlo automaticamente. Dovrai rinnovare il certificato o richiederne uno nuovo. Ulteriori informazioni sul rinnovo dei certificati sono disponibili nella guida alla risoluzione dei problemi relativi al ACM rinnovo gestito dei certificati.

Monitora le API HTTP in API Gateway

Puoi utilizzare CloudWatch metriche e CloudWatch log per monitorare le API HTTP. Combinando log e parametri, puoi registrare gli errori e monitorare le prestazioni dell'API.

> **Note**
>
> L'API Gateway potrebbe non generare log e parametri nei seguenti casi:
>
> - Errori 413 per richiesta troppo grande
> - Errori 429 per troppe richieste
> - Errori serie 400 per richieste inviate a un dominio personalizzato che non dispone del mapping API
> - Errori serie 500 per malfunzionamenti interni

Argomenti

- Monitora le CloudWatch metriche per HTTP APIs in API Gateway
- Configurare la registrazione per le API HTTP in API Gateway

Monitora le CloudWatch metriche per HTTP APIs in API Gateway

Puoi monitorare API l'esecuzione utilizzando CloudWatch, che raccoglie ed elabora i dati grezzi da API Gateway in metriche leggibili. near-real-time Queste statistiche vengono registrate per un periodo di 15 mesi, per permettere l'accesso alle informazioni storiche e offrire una prospettiva migliore sulle prestazioni del servizio o dell'applicazione Web. Per impostazione predefinita, i dati metrici di API Gateway vengono inviati automaticamente in periodi di un CloudWatch minuto. Per monitorare le tue metriche, crea una CloudWatch dashboard per i tuoi. API Per ulteriori informazioni su come creare una CloudWatch dashboard, consulta Creating a CloudWatch dashboard nella Amazon CloudWatch User Guide. Per ulteriori informazioni, consulta What Is Amazon CloudWatch? nella Amazon CloudWatch User Guide.

Le seguenti metriche sono supportate per HTTPAPIs. Puoi anche abilitare metriche dettagliate per scrivere metriche a livello di percorso su Amazon. CloudWatch

Parametro	Descrizione
4xx	Il numero di errori lato client catturati in un dato periodo.
5xx	Il numero di errori lato server catturati in un dato periodo.
Conteggio	Il numero totale di API richieste in un determinato periodo.
IntegrationLatency	Il tempo che intercorre tra il momento in cui API Gateway inoltra una richiesta al backend e il momento in cui riceve una risposta dal backend.
Latenza	Il tempo che intercorre tra il momento in cui API Gateway riceve una richiesta da un client e il momento in cui

Parametro	Descrizione
	restituisce una risposta al client. La latenza include la latenza di integrazione e altri costi generali del API Gateway.
DataProcessed	La quantità di dati elaborati in byte.

È possibile utilizzare le dimensioni nella tabella seguente per API filtrare le metriche di Gateway.

Dimensione	Descrizione
ApiId	Filtra le metriche di API Gateway per un utente API con l'ID specificatoAPI.
ApiId, Fase	Filtra le metriche di API Gateway per uno API stage con l'APIID e l'ID dello stage specificati.
ApiId, Metodo, risorsa, fase	Filtra le metriche del API gateway per un API metodo con l'ID, l'APIID dello stadio, il percorso di risorsa e l'ID del percorso specificati. APIGateway non invierà queste metriche a meno che tu non abbia abilitato esplicita mente le metriche dettaglia te. CloudWatch È possibile farlo richiamando l'UpdateStageazione del API Gateway

Dimensione	Descrizione
	V2 REST API a cui aggiornare la proprietà. `detailedMetricsEnabled` `true` In alternativa, è possibile chiamare il AWS CLI comando update-stage per aggiornare la `DetailedMetricsEnabled` proprietà a. `true` L'abilitazione di tali parametri comporta addebiti aggiuntivi sul tuo account. Per informazioni sui prezzi, consulta la pagina CloudWatchdei prezzi di Amazon.

Configurare la registrazione per le API HTTP in API Gateway

È possibile attivare la registrazione per scrivere i log nei registri. CloudWatch È possibile utilizzare variabili di registrazione per personalizzare il contenuto dei log.

Per attivare la registrazione per un'API HTTP, è necessario effettuare le seguenti operazioni.

1. Assicurarsi che l'utente disponga delle autorizzazioni necessarie per attivare la registrazione.
2. Crea un gruppo di CloudWatch log Logs.
3. Fornisci l'ARN del gruppo di log CloudWatch Logs per una fase della tua API.

Autorizzazioni per attivare la registrazione

Per attivare la registrazione per un'API, l'utente deve disporre delle seguenti autorizzazioni.

Example

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:GetLogEvents",
                "logs:FilterLogEvents"
            ],
            "Resource": "arn:aws:logs:us-east-2:123456789012:log-group:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:PutResourcePolicy",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:CreateLogGroup",
                "logs:DescribeResourcePolicies",
                "logs:GetLogDelivery",
                "logs:ListLogDeliveries"
            ],
            "Resource": "*"
        }
    ]
}
```

Creare un gruppo di log e attivare la registrazione per le API HTTP

È possibile creare un gruppo di log e attivare la registrazione degli accessi utilizzando o il AWS Management Console . AWS CLI

AWS Management Console

1. Creazione di un gruppo di log.

   Per informazioni su come creare un gruppo di log utilizzando la console, consulta Create a Log Group in Amazon CloudWatch Logs User Guide.
2. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
3. Scegliere un'API HTTP.

4. Nella scheda Monitor (Monitoraggio) nel riquadro di navigazione principale, scegliere Logging (Registrazione).
5. Seleziona una fase per attivare la registrazione e scegliere Select (Seleziona).
6. Scegliere Edit (Modifica) per attivare la registrazione degli accessi.
7. Attiva la registrazione degli accessi, inserisci un CloudWatch log e seleziona un formato di log.
8. Selezionare Salva.

AWS CLI

Il AWS CLI comando seguente crea un gruppo di log.

```
aws logs create-log-group --log-group-name my-log-group
```

Per attivare la registrazione, è necessario il nome della risorsa Amazon (ARN) del gruppo di log. *Il formato ARN è arn:aws:logs: region: account-id:log-group:. log-group-name*

Il comando seguente AWS CLI attiva la registrazione per la fase di un'API HTTP. `$default`

```
aws apigatewayv2 update-stage --api-id abcdef \
    --stage-name '$default' \
    --access-log-settings '{"DestinationArn": "arn:aws:logs:region:account-
id:log-group:log-group-name", "Format": "$context.identity.sourceIp - -
 [$context.requestTime] \"$context.httpMethod $context.routeKey $context.protocol\"
 $context.status $context.responseLength $context.requestId"}'
```

Formati di log di esempio

Esempi di formati di log di accesso comuni utilizzati con maggiore frequenza sono disponibili nella console API Gateway ed elencati come segue.

- `CLF` (Common Log Format):

```
$context.identity.sourceIp - - [$context.requestTime] "$context.httpMethod
 $context.routeKey $context.protocol" $context.status $context.responseLength
 $context.requestId $context.extendedRequestId
```

- `JSON`:

```
{ "requestId":"$context.requestId", "ip": "$context.identity.sourceIp",
 "requestTime":"$context.requestTime",
 "httpMethod":"$context.httpMethod","routeKey":"$context.routeKey",
 "status":"$context.status","protocol":"$context.protocol",
 "responseLength":"$context.responseLength", "extendedRequestId":
 "$context.extendedRequestId" }
```

- `XML`:

```
<request id="$context.requestId"> <ip>$context.identity.sourceIp</ip> <requestTime>
$context.requestTime</requestTime> <httpMethod>$context.httpMethod</httpMethod>
 <routeKey>$context.routeKey</routeKey> <status>$context.status</status> <protocol>
$context.protocol</protocol> <responseLength>$context.responseLength</responseLength>
 <extendedRequestId>$context.extendedRequestId</extendedRequestId> </request>
```

- `CSV` (valori separati da virgola):

```
$context.identity.sourceIp,$context.requestTime,$context.httpMethod,
$context.routeKey,$context.protocol,$context.status,$context.responseLength,
$context.requestId,$context.extendedRequestId
```

Personalizzazione dei registri di HTTP API accesso

È possibile utilizzare le seguenti variabili per personalizzare i registri di HTTP API accesso. Per ulteriori informazioni sui log di accesso per HTTPAPIs, consulta. Configurare la registrazione per le API HTTP in API Gateway

Parametro	Descrizione
`$context.accountId`	L'ID dell' AWS account del API proprietario.
`$context.apiId`	L'identificatore che API Gateway assegna al tuo. API
`$context.authorizer.claims.` *`property`*	Una proprietà delle affermazioni restituite dal JSON Web Token (JWT) dopo che il chiamante del metodo è stato autenticato con successo, ad esempio. `$context.authorize`

Parametro	Descrizione
	`r.claims.username` Per ulteriori informazioni, consulta Controlla l'accesso HTTP APIs con gli JWT autorizzatori in API Gateway. **Note** La chiamata di `$context.authorizer.claims` restituisce null.
`$context.authorizer.error`	Il messaggio di errore restituito da un'autorizzazione.
`$context.authorizer.principalId`	Identificazione dell'utente principale restituita da un provider di autorizzazioni Lambda.
`$context.authorizer.` *`property`*	Il valore della coppia chiave-valore specificata della `context` mappa restituita da una funzione di autorizzazione Gateway API Lambda. Ad esempio, se le autorizzazioni restituiscono la mappa `context` seguente: `"context" : { "key": "value", "numKey": 1, "boolKey": true }` chiamando `$context.authorizer.key` viene restituita la stringa `"value"`, chiamando `$context.authorizer.numKey` viene restituito 1 e chiamando `$context.authorizer.boolKey` viene restituito `true`.

Parametro	Descrizione
`$context.awsEndpointRequestId`	L'ID della richiesta dell' AWS endpoint dall'inte stazione or. `x-amz-request-id` `x-amzn-requestId`
`$context.awsEndpointRequestId2`	L'ID della richiesta dell' AWS endpoint dall'inte stazione. `x-amz-id-2`
`$context.customDomain.basePathMatched`	Il percorso di una API mappatura a cui corrisponde una richiesta in arrivo. Applicabile quando un client utilizza un nome di dominio personalizzato per accedere a un. API Ad esempio, se un client invia una richiesta a`https://api.example.com/v1/orders/1234` , e la richiesta corrispon de alla API mappatura con il percorso`v1/orders` , il valore è`v1/orders` . Per ulteriori informazioni, consulta the section called “Mappature API”.
`$context.dataProcessed`	La quantità di dati elaborati in byte.
`$context.domainName`	Il nome di dominio completo utilizzato per richiamare il. API Deve essere lo stesso dell'intestazione `Host` in ingresso.
`$context.domainPrefix`	Prima etichetta di `$context.domainName` .
`$context.error.message`	Una stringa che contiene un messaggio di errore API Gateway.
`$context.error.messageString`	Valore `$context.error.message` tra virgolette, ovvero `"$context.error.message"` .

Parametro	Descrizione
`$context.error.responseType`	Un tipo di `GatewayResponse` . Per ulteriori informazioni, consulta the section called “Metriche” e the section called “Configurazione delle risposte del gateway per la personalizzazione delle risposte agli errori”.
`$context.extendedRequestId`	Equivalente a `$context.requestId` .
`$context.httpMethod`	Il HTTP metodo utilizzato. I valori validi sono: `DELETE`, `GET`, `HEAD`, `OPTIONS`, `PATCH`, `POST` e `PUT`.
`$context.identity.accountId`	L'ID AWS dell'account associato alla richiesta . Supportato per i percorsi che utilizzano IAM l'autorizzazione.
`$context.identity.caller`	Identificatore dell'entità principale dell'intermediario che ha firmato la richiesta. Supportato per i percorsi che utilizzano IAM l'autorizzazione.

Parametro	Descrizione
`$context.identity.cognitoAuthenticationProvider`	Un elenco separato da virgole di tutti i provider di autenticazione Amazon Cognito utilizzati dal chiamante che effettua la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito. Ad esempio, per un'identità di un pool di utenti Amazon Cognito, `cognito-idp.`*`region`*`.amazonaws.com/`*`user_pool_id`*`,cognito-idp.`*`region`*`.amazonaws.com/`*`user_pool_id`*`:CognitoSignIn:`*`token subject claim`* Per informazioni sui provider di autenticazione di Amazon Cognito disponibili, consulta Using Federated Identities nella Amazon Cognito Developer Guide.
`$context.identity.cognitoAuthenticationType`	Tipo di autenticazione Amazon Cognito dell'intermediario da cui proviene la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito. I valori possibili includono `authenticated` per le identità autenticate e `unauthenticated` per le identità non autenticate.
`$context.identity.cognitoIdentityId`	ID identità di Amazon Cognito dell'intermediario da cui proviene la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito.
`$context.identity.cognitoIdentityPoolId`	ID pool di identità di Amazon Cognito dell'intermediario da cui proviene la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito.

Parametro	Descrizione
`$context.identity.principalOrgId`	L'ID organizzazione AWS. Supportato per percorsi che utilizzano l'autorizzazione. IAM
`$context.identity.clientCert.clientCertPem`	Il certificato client PEM con codifica che il client ha presentato durante l'autenticazione reciprocaTLS. Presente quando un client accede a un API utilizzando un nome di dominio personalizzato con attivazione reciproca. TLS
`$context.identity.clientCert.subjectDN`	Nome distinto dell'oggetto del certificato presentato da un client. Presente quando un client accede a un API utilizzando un nome di dominio personalizzato con attivazione reciprocaTLS.
`$context.identity.clientCert.issuerDN`	Nome distinto dell'approvatore del certificato presentato da un cliente. Presente quando un client accede a un API utilizzando un nome di dominio personalizzato con attivazione reciprocaTLS.
`$context.identity.clientCert.serialNumber`	Il numero di serie del certificato. Presente quando un client accede a un API utilizzando un nome di dominio personalizzato con attivazione reciprocaTLS.
`$context.identity.clientCert.validity.notBefore`	La data prima della quale il certificato non è valido. Presente quando un client accede a un API utilizzando un nome di dominio personalizzato con attivazione reciprocaTLS.
`$context.identity.clientCert.validity.notAfter`	La data dopo la quale il certificato non è valido. Presente quando un client accede a un API utilizzando un nome di dominio personalizzato con attivazione reciprocaTLS.

Parametro	Descrizione
`$context.identity.sourceIp`	L'indirizzo IP di origine della TCP connessione immediata che effettua la richiesta all'endpoint API Gateway.
`$context.identity.user`	Identificatore dell'entità principale dell'uten te che sarà autorizzato per l'accesso alle risorse. Supportato per i percorsi che utilizzano l'IAMautorizzazione.
`$context.identity.userAgent`	L'`User-Agent` intestazione del API chiamante .
`$context.identity.userArn`	L'Amazon Resource Name (ARN) dell'uten te effettivo identificato dopo l'autenticazione. Supportato per i percorsi che utilizzano IAM l'autorizzazione. Per ulteriori informazioni, consulta https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html.
`$context.integration.error`	Il messaggio di errore restituito da un'integr azione. Equivalente a `$context.integrationErrorMessage` .
`$context.integration.integrationStatus`	Per l'integrazione del proxy Lambda, il codice di stato restituito dal codice della funzione Lambda di backend AWS Lambda, non dal codice della funzione Lambda.
`$context.integration.latency`	Latenza di integrazione in ms. Equivalente a `$context.integrationLatency` .
`$context.integration.requestId`	L'ID della AWS richiesta dell'endpoint. Equivalente a `$context.awsEndpointRequestId` .

Parametro	Descrizione
`$context.integration.status`	Il codice di stato restituito da un'integrazione. Per le integrazioni proxy Lambda, questo è il codice di stato restituito dal codice della funzione Lambda.
`$context.integrationErrorMessage`	Una stringa contenente un messaggio di errore di integrazione.
`$context.integrationLatency`	Latenza di integrazione in ms.
`$context.integrationStatus`	Per l'integrazione del proxy Lambda, questo parametro rappresenta il codice di stato restituito dalla funzione Lambda di backend AWS Lambda, non dalla funzione Lambda di backend.
`$context.path`	Percorso della richiesta. Ad esempio, `/{stage}/root/child` .
`$context.protocol`	Protocollo della richiesta, ad esempi, `HTTP/1.1`. **Note** APIGateway APIs può accettare richieste HTTP /2, ma API Gateway invia richieste alle integrazioni di backend utilizzando /1.1. HTTP Di conseguenza, il protocollo di richiesta viene registrato come HTTP /1.1 anche se un client invia una richiesta che utilizza /2. HTTP
`$context.requestId`	L'ID assegnato da API Gateway alla richiesta. API

Parametro	Descrizione
`$context.requestTime`	L'ora della richiesta in CLFformato -formattato (). `dd/MMM/yyyy:HH:mm:ss +-hhmm`
`$context.requestTimeEpoch`	Ora della richiesta in formato Unix epoch.
`$context.responseLatency`	Latenza della risposta in ms.
`$context.responseLength`	Lunghezza del payload della risposta in byte.
`$context.routeKey`	La chiave di percorso della API richiesta, ad esempio. `/pets`
`$context.stage`	La fase di distribuzione della API richiesta (ad esempio, `beta` o`prod`).
`$context.status`	Stato della risposta del metodo.

Risoluzione dei problemi relativi alle API HTTP in API Gateway

Negli argomenti seguenti vengono forniti suggerimenti per la risoluzione dei problemi relativi a errori e problemi che potrebbero verificarsi durante l'utilizzo di API HTTP.

Argomenti

- Risoluzione dei problemi relativi alle integrazione Lambda di API HTTP
- Risoluzione dei problemi relativi ai provider di autorizzazioni JWT per API HTTP

Risoluzione dei problemi relativi alle integrazione Lambda di API HTTP

Di seguito viene fornita una consulenza per la risoluzione dei problemi relativi a errori e problemi che potrebbero verificarsi quando si utilizza AWS Lambda integrazioni con API HTTP.

Problema: la mia API con un'integrazione Lambda restituisce `{"message":"Internal Server Error"}`

Per risolvere l'errore interno del server, aggiungere la `$context.integrationErrorMessage` variabile di registrazione al formato di registro e visualizzare i registri dell'API HTTP. Per raggiungere questo obiettivo, effettuare le seguenti operazioni:

Per creare un gruppo di log utilizzando il AWS Management Console

1. Aprire la CloudWatch console all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Scegliere Gruppi di log.
3. Scegliere Crea gruppo di log.
4. Immettere un nome di gruppo di log, quindi scegliere Crea.
5. Prendere nota del nome risorsa Amazon (ARN) per il tuo gruppo di log. *`Il formato ARN è arn:aws:logs: region: account-id:log-group:. log-group-name`* È necessario il gruppo di log ARN per abilitare la registrazione degli accessi per la propria API HTTP.

Per aggiungere la variabile di registrazione **`$context.integrationErrorMessage`**

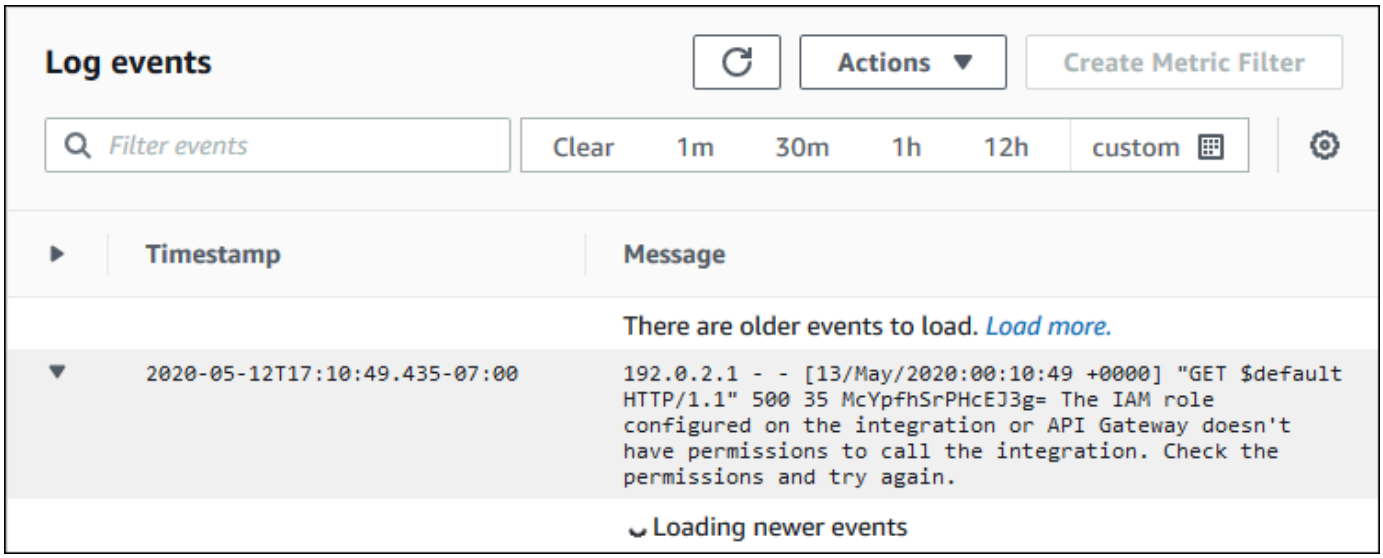
1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere l'API HTTP.
3. In Monitor (Monitora), scegliere Logging (Registrazione).
4. Selezionare una fase della tua API.
5. Scegliere Modifica, quindi abilitare la registrazione degli accessi.
6. Per Log destination (Destinazione dei log), immettere l'ARN per il gruppo di log creato nel passaggio precedente.
7. Per Formato registro, scegliere CLF. API Gateway crea un formato di registro di esempio.
8. Aggiungere `$context.integrationErrorMessage` alla fine del formato di registro.
9. Selezionare Salva.

Per visualizzare i log delle API

1. Generare i log Utilizzare un browser o `curl` per richiamare la tua API.

```
$curl https://api-id.execute-api.us-west-2.amazonaws.com/route
```

2. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
3. Scegliere l'API HTTP.
4. In Monitor (Monitora), scegliere Logging (Registrazione).
5. Selezionare la fase dell'API per la quale è stata abilitata la registrazione.
6. Scegli Visualizza i log in CloudWatch.
7. Scegliere il flusso di log più recente per visualizzare i log dell'API HTTP.
8. La voce di registro dovrebbe essere simile alla seguente:



Poiché è stato aggiunto `$context.integrationErrorMessage` al formato di registro, viene visualizzato un messaggio di errore nei nostri registri che riassume il problema.

I registri potrebbero includere un messaggio di errore diverso che indica che si è verificato un problema con il codice della funzione Lambda. In tal caso, controllare il codice della funzione Lambda e verificare che la funzione Lambda restituisca una risposta nel formato richiesto.
Se i log non includono messaggi di errore, aggiungere `$context.error.message` e `$context.error.responseType` al formato del log per ottenere ulteriori informazioni per la risoluzione dei problemi.

In questo caso, i registri mostrano che API Gateway non disponeva delle autorizzazioni necessarie per richiamare la funzione Lambda.

Quando si crea un'integrazione Lambda nella console API Gateway, le autorizzazioni vengono configurate automaticamente per richiamare la funzione Lambda. Quando crei un'integrazione Lambda utilizzando, o un SDK AWS CLI AWS CloudFormation, devi concedere le autorizzazioni

ad API Gateway per richiamare la funzione. I seguenti AWS CLI comandi di esempio concedono l'autorizzazione a diversi percorsi API HTTP per richiamare una funzione Lambda.

Example Esempio: per la **`$default`** fase e il **`$default`** percorso di un'API HTTP

```
aws lambda add-permission \
    --function-name my-function \
    --statement-id apigateway-invoke-permissions \
    --action lambda:InvokeFunction \
    --principal apigateway.amazonaws.com \
    --source-arn "arn:aws:execute-api:us-west-2:123456789012:api-id/\$default/\
$default"
```

Example Esempio: per la **`prod`** fase e il **`test`** percorso di un'API HTTP

```
aws lambda add-permission \
    --function-name my-function \
    --statement-id apigateway-invoke-permissions \
    --action lambda:InvokeFunction \
    --principal apigateway.amazonaws.com \
    --source-arn "arn:aws:execute-api:us-west-2:123456789012:api-id/prod/*/test"
```

Conferma la policy della funzione nella scheda Permissions (Autorizzazioni) della console Lambda.

Prova a richiamare nuovamente la tua API. Deve essere visualizzata la risposta della funzione Lambda.

Risoluzione dei problemi relativi ai provider di autorizzazioni JWT per API HTTP

Di seguito viene fornita una consulenza per la risoluzione dei problemi relativi agli errori che potrebbero verificarsi quando si utilizzano i provider di autorizzazioni JSON Web Token (JWT) con API HTTP.

Problema: la mia API restituisce **`401 {"message":"Unauthorized"}`**

Controlla l'intestazione `www-authenticate` nella risposta dell'API.

Il comando seguente utilizza `curl` per inviare una richiesta a un'API con un autorizzatore JWT che utilizza `$request.header.Authorization` come origine identità.

```
$curl -v -H "Authorization: token" https://api-id.execute-api.us-
west-2.amazonaws.com/route
```

La risposta dell'API include un'intestazione `www-authenticate`.

```
...
< HTTP/1.1 401 Unauthorized
< Date: Wed, 13 May 2020 04:07:30 GMT
< Content-Length: 26
< Connection: keep-alive
< www-authenticate: Bearer scope="" error="invalid_token" error_description="the token
 does not have a valid audience"
< apigw-requestid: Mc7UVioPPHcEKPA=
<
* Connection #0 to host api-id.execute-api.us-west-2.amazonaws.com left intact
{"message":"Unauthorized"}}
```

In questo caso, l'intestazione `www-authenticate` mostra che il token non è stato emesso per un pubblico valido. Per consentire ad API Gateway di autorizzare una richiesta, la richiesta `aud` o `client_id` di JWT deve corrispondere a una delle voci destinatario configurate per il provider di autorizzazioni. API Gateway viene convalidato `client_id` solo se non `aud` è presente. Quando entrambi `aud` `client_id` sono presenti, API Gateway valuta. `aud`

Puoi anche decodificare un JWT e verificare che corrisponda all'emittente, al pubblico e agli ambiti richiesti dall'API. Il sito web jwt.io può eseguire il debug di JWT nel browser. Anche OpenID Foundation mantiene un elenco di librerie per l'utilizzo con JWT.

Per ulteriori informazioni sui provider di autorizzazioni JWT, consulta Controlla l'accesso HTTP APIs con gli JWT autorizzatori in API Gateway.

APIGateway WebSocket APIs

Un WebSocket API in API Gateway è una raccolta di WebSocket percorsi integrati con HTTP endpoint di backend, funzioni Lambda o altri servizi. AWS Puoi utilizzare le funzionalità di API Gateway per aiutarti in tutti gli aspetti del API ciclo di vita, dalla creazione al monitoraggio della produzione. APIs

APII gateway WebSocket APIs sono bidirezionali. Un client può inviare messaggi a un servizio e i servizi possono inviare messaggi ai client in modo indipendente. Questo comportamento bidirezionale consente interazioni client/servizio più ricche perché i servizi possono inviare dati ai client senza richiedere ai clienti di effettuare una richiesta esplicita. WebSocket APIsvengono spesso utilizzati in applicazioni in tempo reale come applicazioni di chat, piattaforme di collaborazione, giochi multiplayer e piattaforme di trading finanziario.

Per un'app di esempio con cui iniziare, consulta Tutorial: crea un'app di WebSocket chat con a WebSocket API, Lambda e DynamoDB.

In questa sezione, puoi imparare a sviluppare, pubblicare, proteggere e monitorare l' WebSocket APIsutilizzo di API Gateway.

Argomenti

- Panoramica di WebSocket APIs in API Gateway
- Sviluppa WebSocket APIs in API Gateway
- Pubblica WebSocket API che i clienti possono richiamare
- Proteggi le tue WebSocket API in API Gateway
- Monitora WebSocket le API in API Gateway

Panoramica di WebSocket APIs in API Gateway

In API Gateway è possibile creare un frontend WebSocket API come stateful per un AWS servizio (come Lambda o DynamoDB) o per un endpoint. HTTP WebSocket APIRichiama il backend in base al contenuto dei messaggi che riceve dalle app client.

A differenza di a RESTAPI, che riceve e risponde alle richieste, a WebSocket API supporta la comunicazione bidirezionale tra le app client e il backend. Il back-end può inviare messaggi di callback a client connessi.

Nella tua app WebSocket API, i JSON messaggi in arrivo vengono indirizzati alle integrazioni di backend in base ai percorsi che configuri. (I non JSON messaggi vengono indirizzati a un `$default` percorso configurato dall'utente).

Una route include una chiave di route, che è il valore previsto dopo che un'espressione di selezione della route è stata valutata. `routeSelectionExpression`È un attributo definito a API livello. Specifica una JSON proprietà che dovrebbe essere presente nel payload del messaggio. Per ulteriori informazioni sulle espressioni di selezione della route, consulta the section called “”.

Ad esempio, se i JSON messaggi contengono una `action` proprietà e si desidera eseguire azioni diverse in base a questa proprietà, l'espressione di selezione del percorso potrebbe essere. `${request.body.action}` La tabella di routing specifica quale operazione eseguire abbinando il valore della proprietà `action` ai valori delle chiavi route personalizzate definiti nella tabella.

Usa i percorsi per un WebSocket API

Esistono tre route predefinite che possono essere utilizzate: `$connect`, `$disconnect` e `$default`. Inoltre, puoi creare route personalizzate.

- APIGateway chiama la `$connect` route quando viene avviata una connessione persistente tra il client e a. WebSocket API
- APIGateway chiama il `$disconnect` percorso quando il client o il server si disconnette da. API
- APIGateway chiama una route personalizzata dopo che l'espressione di selezione della rotta è stata valutata rispetto al messaggio se viene trovata una route corrispondente; la corrispondenza determina quale integrazione viene richiamata.
- APIGateway chiama la `$default` route se l'espressione di selezione della rotta non può essere valutata rispetto al messaggio o se non viene trovata alcuna route corrispondente.

Per ulteriori informazioni sulle route `$connect` e `$disconnect`, consulta the section called “Gestisci gli utenti connessi e le app client”.

Per ulteriori informazioni sulla route `$default` e le route personalizzate, consulta the section called “Richiama la tua integrazione con il backend”.

Invia dati alle app client connesse

I servizi di back-end possono inviare dati alle app client connesse. Puoi inviare dati effettuando le seguenti operazioni:

- Utilizzate un'integrazione per inviare una risposta, che viene restituita al client da una risposta di percorso che avete definito.
- Puoi usare il `@connections` API per inviare una POST richiesta. Per ulteriori informazioni, consulta the section called “Utilizzo di comandi `@connections` nel servizio di back-end”.

Gestisci gli utenti connessi e le app client: **`$connect`** e **`$disconnect`** i percorsi

La sezione seguente descrive come utilizzare i `$disconnect` percorsi `$connect` e per WebSocket API.

Argomenti

- Instradamento $connect
- Passaggio delle informazioni di connessione dalla route $connect
- Instradamento $disconnect

Instradamento **`$connect`**

Le app client si connettono al tuo WebSocket API inviando una richiesta di WebSocket aggiornamento. Se la richiesta ha esito positivo, la route `$connect` viene eseguita mentre la connessione viene stabilita.

Poiché la WebSocket connessione è una connessione con stato, puoi configurare l'autorizzazione solo sulla `$connect` route. `AuthN`/`AuthZ`verrà eseguito solo al momento della connessione.

Finché l'esecuzione dell'integrazione associata alla route `$connect` non viene completata, la richiesta di aggiornamento è in sospeso e la connessione effettiva non verrà stabilita. Se la richiesta `$connect` ha esito negativo (ad esempio, a causa di un errore `AuthN`/`AuthZ` o errore di integrazione), la connessione non verrà stabilita.

> **Note**
>
> Se l'autorizzazione ha esito negativo su `$connect`, la connessione non verrà stabilita e il client riceverà una risposta `401` o `403`.

La configurazione di un'integrazione per `$connect` è facoltativa. È opportuno valutare la configurazione di un'integrazione `$connect` se:

- Si desidera consentire ai client di specificare sottoprotocolli utilizzando il campo `Sec-WebSocket-Protocol`. Per il codice di esempio, consulta Configura una `$connect` route che richiede un WebSocket sottoprotocollo.
- Desideri ricevere una notifica in caso di connessione dei client.
- Desideri limitare le connessioni o controllare chi si connette.
- Vuoi che il tuo backend invii messaggi ai clienti utilizzando un callbackURL.
- Desideri archiviare ogni ID connessione e altre informazioni in un database (ad esempio Amazon DynamoDB).

Passaggio delle informazioni di connessione dalla route **`$connect`**

È possibile utilizzare integrazioni proxy e non proxy per passare informazioni dall'instradamento `$connect` a un database o a un altro Servizio AWS.

Per passare le informazioni di connessione utilizzando un'integrazione proxy

È possibile accedere alle informazioni di connessione da un'integrazione proxy Lambda nell'evento. Usa un'altra AWS Lambda funzione Servizio AWS o per postare sulla connessione.

La seguente funzione Lambda mostra come utilizzare l'oggetto `requestContext` per registrare l'ID di connessione, il nome di dominio, il nome della fase e le stringhe di query.

Node.js

```
 export const handler = async(event, context) => {
    const connectId = event["requestContext"]["connectionId"]
    const domainName = event["requestContext"]["domainName"]
    const stageName = event["requestContext"]["stage"]
    const qs = event['queryStringParameters']
    console.log('Connection ID: ', connectId, 'Domain Name: ', domainName, 'Stage
 Name: ', stageName, 'Query Strings: ', qs )
    return {"statusCode" : 200}
};
```

Python

```
import json
```

```
import logging
logger = logging.getLogger()
logger.setLevel("INFO")


def lambda_handler(event, context):
    connectId = event["requestContext"]["connectionId"]
    domainName = event["requestContext"]["domainName"]
    stageName = event["requestContext"]["stage"]
    qs = event['queryStringParameters']
    connectionInfo = {
        'Connection ID': connectId,
        'Domain Name': domainName,
        'Stage Name': stageName,
        'Query Strings': qs}
    logging.info(connectionInfo)
    return {"statusCode": 200}
```

Per passare informazioni di connessione utilizzando un'integrazione proxy

- È possibile accedere alle informazioni di connessione da un'integrazione non proxy. Imposta la richiesta di integrazione e fornisci un modello di WebSocket API richiesta. Il seguente modello di mappatura Velocity Template Language (VTL) fornisce una richiesta di integrazione. Questa richiesta invia i seguenti dettagli a un'integrazione non proxy:

  - ID connessione
  - Nome dominio
  - Nome fase
  - Path
  - Headers
  - Stringhe di query

  Questa richiesta invia l'ID di connessione, il nome di dominio, il nome della fase i percorsi, le intestazioni e le stringhe di query a un'integrazione non proxy.

```
{
    "connectionId": "$context.connectionId",
    "domain": "$context.domainName",
```

```
    "stage": "$context.stage",
    "params": "$input.params()"
}
```

Per ulteriori informazioni sulla configurazione delle trasformazioni di dati, consulta the section called “Trasformazioni dei dati”.

Per completare la richiesta di integrazione, impostare `StatusCode: 200` per la risposta di integrazione. Per ulteriori informazioni sulla configurazione di una risposta di integrazione, consulta Imposta una risposta di integrazione utilizzando la console API Gateway.

Instradamento **`$disconnect`**

La route `$disconnect` viene eseguita dopo la chiusura della connessione.

La connessione può essere chiusa dal server o dal client. Poiché la connessione è già chiusa quando viene eseguita, `$disconnect` è un evento best-effort. APIGateway farà del suo meglio per fornire l'`$disconnect`evento necessario all'integrazione, ma non può garantire la consegna.

Il backend può avviare la disconnessione utilizzando. `@connections` API Per ulteriori informazioni, consulta the section called “Utilizzo di comandi `@connections` nel servizio di back-end”.

Richiama l'integrazione del backend con **`$default`** Route e i percorsi personalizzati in Gateway API

La sezione seguente descrive come richiamare l'integrazione del backend utilizzando la `$default` route o una route personalizzata per a. WebSocket API

Argomenti

- Utilizzo di instradamenti per elaborare messaggi
- Instradamento $default
- Instradamenti personalizzati
- Utilizzo WebSocket API delle integrazioni API Gateway per connettersi alla logica aziendale
- Differenze importanti tra e WebSocket APIs REST APIs

Utilizzo di instradamenti per elaborare messaggi

In API Gateway WebSocket APIs, i messaggi possono essere inviati dal client al servizio di backend e viceversa. A differenza HTTP del modello di richiesta/risposta, WebSocket nel backend è possibile inviare messaggi al client senza che il client intraprenda alcuna azione.

I messaggi possono essere JSON o meno. JSON Tuttavia, solo JSON i messaggi possono essere indirizzati a integrazioni specifiche in base al contenuto del messaggio. I messaggi diversi dai JSON messaggi vengono passati al backend tramite il percorso. `$default`

> **Note**
>
> APIGateway supporta payload di messaggi fino a 128 KB con una dimensione massima dei frame di 32 KB. Se un messaggio supera i 32 KB, occorre suddividerlo in più frame, ciascuno di 32 KB o più piccolo. Se si riceve un messaggio (o frame) di dimensioni maggiori, la connessione viene chiusa con codice 1009.
> Al momento i payload binari non sono supportati. Se si riceve un frame binario, la connessione viene chiusa con codice 1003. Tuttavia, è possibile convertire i payload binari in testo. Consulta the section called “Tipi di supporti binari”.

Con WebSocket APIs in API Gateway, JSON i messaggi possono essere instradati per eseguire un servizio di backend specifico basato sul contenuto dei messaggi. Quando un client invia un messaggio tramite la sua WebSocket connessione, ciò si traduce in una richiesta di routing a. WebSocket API La richiesta verrà abbinata alla route con la chiave di route corrispondente in API Gateway. È possibile impostare una richiesta di route per WebSocket API a nella console API Gateway AWS CLI, utilizzando o utilizzando un AWS SDK.

> **Note**
>
> Infine AWS SDKs, puoi creare percorsi prima o dopo aver creato le integrazioni. AWS CLI Attualmente la console non supporta il riutilizzo di integrazioni, perciò è necessario creare la route prima e quindi creare l'integrazione per tale route.

È possibile configurare API Gateway per eseguire la convalida di una richiesta di percorso prima di procedere con la richiesta di integrazione. Se la convalida fallisce, API Gateway fallisce la richiesta

senza chiamare il backend, invia al client una risposta del `"Bad request body"` gateway simile alla seguente e pubblica i risultati della convalida in Logs: CloudWatch

```
{"message" : "Bad request body", "connectionId": "{connectionId}", "messageId":
 "{messageId}"}
```

Ciò riduce le chiamate non necessarie al backend e ti consente di concentrarti sugli altri requisiti del tuo. API

Puoi anche definire una risposta di routing per i tuoi percorsi per abilitare API la comunicazione bidirezionale. Una risposta di instradamento descrive quali dati verranno inviati al client al termine dell'integrazione di una particolare route. Non è necessario definire una risposta per una route se, ad esempio, desideri che un client invii messaggi al back-end senza ricevere una risposta (comunicazione unidirezionale). Tuttavia, se non fornisci una risposta di routing, API Gateway non invierà alcuna informazione sul risultato dell'integrazione ai tuoi clienti.

Instradamento **`$default`**

Ogni API Gateway WebSocket API può avere un `$default` percorso. Si tratta di un valore di instradamento speciale che può essere utilizzato nei seguenti modi:

- Puoi usarlo insieme a chiavi di route definite per specificare una route di "fallback", ad esempio un'integrazione fittizia generica che restituisce un determinato messaggio di errore, per i messaggi in ingresso che non corrispondono ad alcuna delle chiavi di route definite.
- Puoi utilizzarlo senza alcuna chiave route definita, per specificare un modello di proxy che delega l'instradamento a un componente di back-end.
- È possibile utilizzarlo per specificare un percorso per carichi diversi dai JSON payload.

Instradamenti personalizzati

Se desideri invocare un'integrazione specifica in base al contenuto del messaggio, puoi farlo creando una route personalizzata.

Una route personalizzata utilizza una chiave route e l'integrazione specificati dall'utente. Quando un messaggio in arrivo contiene una JSON proprietà e tale proprietà restituisce un valore che corrisponde al valore della chiave della route, API Gateway richiama l'integrazione. (Per ulteriori informazioni, consulta the section called “Panoramica di WebSocket APIs”.)

Ad esempio, supponiamo che tu voglia creare un'applicazione chat room. È possibile iniziare creando un'espressione di selezione del percorso WebSocket API la cui espressione è. `$request.body.action` Potresti quindi definire due route: `joinroom` e `sendmessage`. Un'app client potrebbe richiamare la route `joinroom` inviando un messaggio del tipo seguente:

```
{"action":"joinroom","roomname":"developers"}
```

E potrebbe richiamare la route `sendmessage` inviando un messaggio del tipo seguente:

```
{"action":"sendmessage","message":"Hello everyone"}
```

Utilizzo WebSocket API delle integrazioni API Gateway per connettersi alla logica aziendale

Dopo aver impostato un percorso per un API Gateway WebSocket API, devi specificare l'integrazione che desideri utilizzare. Come nel caso di una route, per la quale è possibile avere una richiesta di instradamento e una risposta di instradamento, un'integrazione può avere una richiesta di integrazione e una risposta di integrazione. Una richiesta di integrazione contiene le informazioni previste dal back-end per elaborare la richiesta proveniente dal client. Una risposta di integrazione contiene i dati che il backend restituisce a API Gateway e che possono essere utilizzati per costruire un messaggio da inviare al client (se è definita una risposta di routing).

Per ulteriori informazioni sulla configurazione di integrazioni, consulta the section called “Integrazioni”.

Differenze importanti tra e WebSocket APIs REST APIs

Le integrazioni per WebSocket APIs sono simili alle integrazioni per RESTAPIs, ad eccezione delle seguenti differenze:

- Attualmente, nella console API Gateway è necessario creare prima una route e poi creare un'integrazione come destinazione di quella route. Tuttavia, alla API fineCLI, puoi creare percorsi e integrazioni indipendentemente, in qualsiasi ordine.
- Puoi utilizzare una singola integrazione per più route. Ad esempio, se disponi di un set di operazioni tra loro strettamente correlate, puoi fare in modo che tutte queste route vadano in una singola funzione Lambda. Anziché definire più volte i dettagli dell'integrazione, puoi specificarla una sola volta e assegnarla a ciascuna delle route correlate.

> **Note**
>
> Attualmente la console non supporta il riutilizzo di integrazioni, perciò è necessario creare la route prima e quindi creare l'integrazione per tale route.
> Infine AWS SDKs, puoi riutilizzare un'integrazione impostando la destinazione del percorso su un valore di"`integrations/{integration-id}`", dove *`{integration-id}`*" è l'ID univoco dell'integrazione da associare al percorso. AWS CLI

- APIGateway fornisce più espressioni di selezione che puoi utilizzare nei tuoi percorsi e nelle tue integrazioni. Non è necessario basarsi sul tipo di contenuto per selezionare un modello di input o una mappatura di output. Come per le espressioni di selezione delle rotte, è possibile definire un'espressione di selezione da valutare da API Gateway per scegliere l'elemento giusto. Tutte verranno ripristinati al modello `$default` se non viene trovato un modello corrispondente.
  - Nelle richieste di integrazione, l'espressione di selezione del modello supporta `$request.body.`*`<json_path_expression>`* e valori statici.
  - Nelle risposte di integrazione, l'espressione di selezione del modello supporta `$request.body.`*`<json_path_expression>`*, `$integration.response.statuscode`, `$integration.response.header.`*`<headerName>`* e valori statici.

Nel HTTP protocollo, in cui le richieste e le risposte vengono inviate in modo sincrono, la comunicazione è essenzialmente unidirezionale. Nel WebSocket protocollo, la comunicazione è bidirezionale. Le risposte sono asincrone e non vengono necessariamente ricevute dal client nello stesso ordine di invio dei messaggi del client. Inoltre, il back-end può inviare messaggi al client.

> **Note**
>
> Per un percorso configurato per l'uso `AWS_PROXY` o l'`LAMBDA_PROXY`integrazione, la comunicazione è unidirezionale e API Gateway non trasmetterà automaticamente la risposta del backend alla risposta di routing. Ad esempio, in caso dell'integrazione `LAMBDA_PROXY`, il corpo restituito dalla funzione Lambda non verrà restituito al client. Se desideri che il client riceva risposte di integrazione, devi definire una risposta di instradamento per rendere possibile la comunicazione bidirezionale.

WebSocket espressioni di selezione

APIGateway utilizza le espressioni di selezione per valutare il contesto di richiesta e risposta e produrre una chiave. La chiave viene quindi utilizzata per selezionare tra un insieme di valori possibili, in genere forniti dallo API sviluppatore. L'esatto set di variabili supportate varierà a seconda del tipo di espressione. Ogni espressione viene descritta in modo dettagliato di seguito.

Per tutte le espressioni, il linguaggio segue lo stesso set di regole:

- Una variabile ha il prefisso `"$"`.
- Le parentesi graffe possono essere utilizzate per definire i limiti delle variabili in modo esplicito, ad esemp., `"${request.body.version}-beta"`.
- Sono supportate numerose variabili, ma la valutazione viene eseguita una sola volta (non esiste la valutazione ricorsiva).
- Il segno di dollaro ($) può essere preceduto dal carattere di escape `"\"`. Questa prassi è molto utile quando si definisce un'espressione associata alla chiave riservata `$default`, ad esempio `"\$default"`.
- In alcuni casi, un formato di modello è obbligatorio. In questo caso, l'espressione deve essere racchiusa tra barre (`"/"`), ad esempio `"/2\d\d/"` per corrispondere ai codici di stato 2*XX*.

Argomenti

- Espressioni di selezione della risposta di instradamento
- APIespressioni di selezione chiave
- APImappatura delle espressioni di selezione
- WebSocketriepilogo delle espressioni di selezione

Espressioni di selezione della risposta di instradamento

Una risposta di instradamento viene utilizzata per modellare una risposta dal back-end al client. Infatti WebSocket APIs, una risposta di percorso è facoltativa. Una volta definita, segnala a API Gateway che deve restituire una risposta a un client dopo aver ricevuto un WebSocket messaggio.

La valutazione dell'espressione di selezione della risposta di instradamento produce una chiave di risposta di instradamento. Alla fine, questa chiave verrà utilizzata per scegliere tra quelle `RouteResponses`associate aAPI. Tuttavia, al momento è supportata solo la chiave `$default`.

APIespressioni di selezione chiave

Questa espressione viene valutata quando il servizio determina che la richiesta specificata deve procedere solo se il client fornisce una APIchiave valida.

Attualmente, i soli due valori supportati sono `$request.header.x-api-key` e `$context.authorizer.usageIdentifierKey`.

APImappatura delle espressioni di selezione

Questa espressione viene valutata per determinare quale API fase viene selezionata quando viene effettuata una richiesta utilizzando un dominio personalizzato.

Attualmente, l'unico valore supportato è `$request.basepath`.

WebSocketriepilogo delle espressioni di selezione

La tabella seguente riassume i casi d'uso delle espressioni di selezione in WebSocketAPIs:

Espressione di selezione	Restituisce la chiave per	Note	Esempio di caso d'uso
`Api.RouteSelectionExpression`	`Route.RouteKey`	`$defaul` chiave è supporta a come instradar ento catch-all.	WebSocket Indirizza i messaggi in base al contesto della richiesta di un client.
`Route.ModelSelecti`	Chiave per `Route.RequestModels`	Facoltati vo.	Convalida dinamica della

Espressione di selezione	Restituisce la chiave per	Note	Esempio di caso d'uso
`onExpression`		Se fornita per un'integrazione non proxy, viene effettuata la convalida del modello. `$default` chiave è supportata come catch-all.	richiesta all'interno dello stesso instradamento.

Espressione di selezione	Restituisce la chiave per	Note	Esempio di caso d'uso
`Integration.TemplateSelectionExpression`	Chiave per `Integration.RequestTemplates`	Facoltativo. Può essere fornita per un'integrazione non proxy allo scopo di manipolare i payload in entrata. `${reque .body.j nPath}` supporta i e valori statici. `$defaul` chiave è supporta a	Manipolazione della richiesta dell'intermediario in base alle proprietà dinamiche della richiesta.

Espressione di selezione	Restituisce la chiave per	Note	Esempio di caso d'uso
		come catch-all.	

Espressione di selezione	Restituisce la chiave per	Note	Esempio di caso d'uso
`Integration.IntegrationResponseSelectionExpression`	`IntegrationResponse.IntegrationResponseKey`	Facoltativo. Può essere fornita per un'integrazione non proxy. Funge da pattern match per i messaggi di errore (da Lambda) o i codici di stato (dalle HTTP integrazioni). `$default` necessaria la	Manipolazione della risposta del back-end. Scegliere l'azione da eseguire in base alla risposta dinamica del back-end, ad esempio la gestione distinta di determinati errori.

Espressione di selezione	Restituisce la chiave per	Note	Esempio di caso d'uso
		chiave per far sì che le integrazioni non proxy agiscano da catch-all per le risposte con esito positivo.	

Espressione di selezione	Restituisce la chiave per	Note	Esempio di caso d'uso
`IntegrationResponse.TemplateSelectionExpression`	Chiave per `IntegrationResponse.ResponseTemplates`	Facoltativo. Può essere fornita per un'integrazione non proxy. La chiave $default è supportata.	In alcuni casi, una proprietà dinamica della risposta potrebbe imporre trasformazioni differenti all'interno dello stesso instradamento e dell'integrazione associata. `${request.body.jsonPath}`, `${integration.response.statuscode}`, `${integra`

Espressione di selezione	Restituisce la chiave per	Note	Esempio di caso d'uso
			`tion.response.header.headerName}` , `${integration.response.multivalueheader.headerName}` , Sono supportati , , , e valori statici.
			`$defaultLa` chiave è supportata come catch-all.

Espressione di selezione	Restituisce la chiave per	Note	Esempio di caso d'uso
`Route.RouteResponseSelectionExpression`	`RouteResponse.RouteResponseKey`	Deve essere fornito per avviare una comunicazione bidirezionale per un percorso WebSocl Attualmente, questo valore è limitato solo a `$defaul`	
`RouteResponse.ModelSelectionExpression`	Chiave per `RouteResponse.RequestModels`	Attualmente non è supportata.	

Sviluppa WebSocket APIs in API Gateway

Questa sezione fornisce dettagli sulle funzionalità di API Gateway di cui hai bisogno durante lo sviluppo del tuo API GatewayAPIs.

Durante lo sviluppo del tuo API GatewayAPI, decidi in base a una serie di caratteristiche del tuoAPI. Queste caratteristiche dipendono dal caso d'uso del tuoAPI. Ad esempio, potresti voler consentire solo a determinati clienti di chiamare il tuoAPI, oppure potresti voler che il servizio sia disponibile per tutti. Potresti volere una API chiamata per eseguire una funzione Lambda, effettuare una query sul database o chiamare un'applicazione.

Argomenti

- Crea WebSocket API in API Gateway
- Crea percorsi per WebSocket APIs in API Gateway
- Controlla e gestisci l'accesso alle WebSocket API in API Gateway
- Integrazioni per WebSocket API in API Gateway
- Richiedi la convalida delle API in WebSocket API Gateway
- Trasformazioni dei dati per le API in WebSocket API Gateway
- Tipi di supporti binari per le WebSocket API in API Gateway
- Richiama le API WebSocket

Crea WebSocket API in API Gateway

Puoi creare un' WebSocket API nella console API Gateway, utilizzando il comando AWS CLI create-api o utilizzando il `CreateApi` comando in un AWS SDK. Le seguenti procedure mostrano come creare una nuova API. WebSocket

> **Note**
>
> WebSocket Le API supportano solo TLS 1.2 e TLS 1.3. Le versioni precedenti di TLS non sono supportate.

Crea un'API utilizzando i comandi WebSocket AWS CLI

La creazione di un' WebSocket API utilizzando AWS CLI richiede la chiamata del comando create-api come illustrato nell'esempio seguente, che crea un'API con l'espressione di selezione del `$request.body.action` percorso:

```
aws apigatewayv2 --region us-east-1 create-api --name "myWebSocketApi3" --protocol-type
 WEBSOCKET --route-selection-expression '$request.body.action'
```

Output di esempio:

```
{
    "ApiKeySelectionExpression": "$request.header.x-api-key",
    "Name": "myWebSocketApi3",
    "CreatedDate": "2018-11-15T06:23:51Z",
    "ProtocolType": "WEBSOCKET",
    "RouteSelectionExpression": "'$request.body.action'",
    "ApiId": "aabbccddee"
}
```

Creare un' WebSocketAPI utilizzando la console API Gateway

Puoi creare un' WebSocket API nella console scegliendo il WebSocket protocollo e assegnando un nome all'API.

> **Important**
>
> Dopo aver creato l'API, non puoi modificare il protocollo scelto per la stessa. Non è possibile convertire un' WebSocket API in un'API REST o viceversa.

Per creare un' WebSocket API utilizzando la console API Gateway

1. Accedere alla console API Gateway e scegliere Create API (Crea API).
2. In WebSocket API, scegli Build. Sono supportati solo gli endpoint regionali.
3. In Nome API immetti il nome dell'API.
4. Per Espressione di selezione dell'instradamento immetti un valore. Ad esempio, `$request.body.action`.

Per ulteriori informazioni sulle espressioni di selezione della route, consulta the section called “”.

5. Esegui una di queste operazioni:
   - Scegli Crea API vuota per creare un'API senza instradamenti.
   - Scegli Successivo per collegare gli instradamenti all'API.

   Puoi collegare gli instradamenti dopo aver creato l'API.

Crea percorsi per WebSocket APIs in API Gateway

Nella tua app WebSocket API, i JSON messaggi in arrivo vengono indirizzati alle integrazioni di backend in base ai percorsi che configuri. (I non JSON messaggi vengono indirizzati a un `$default` percorso configurato dall'utente.)

Una route include una chiave di route, che è il valore previsto dopo che un'espressione di selezione della route è stata valutata. `routeSelectionExpression`È un attributo definito a API livello. Specifica una JSON proprietà che dovrebbe essere presente nel payload del messaggio. Per ulteriori informazioni sulle espressioni di selezione della route, consulta the section called “”.

Ad esempio, se i JSON messaggi contengono una `action` proprietà e si desidera eseguire azioni diverse in base a tale proprietà, l'espressione di selezione del percorso potrebbe essere. `${request.body.action}` La tabella di routing specifica quale operazione eseguire abbinando il valore della proprietà `action` ai valori delle chiavi route personalizzate definiti nella tabella.

Esistono tre route predefinite che possono essere utilizzate: `$connect`, `$disconnect` e `$default`. Inoltre, puoi creare route personalizzate.

- APIGateway chiama la `$connect` route quando viene avviata una connessione persistente tra il client e a. WebSocket API
- APIGateway chiama il `$disconnect` percorso quando il client o il server si disconnette da. API
- APIGateway chiama una route personalizzata dopo che l'espressione di selezione della rotta è stata valutata rispetto al messaggio se viene trovata una route corrispondente; la corrispondenza determina quale integrazione viene richiamata.
- APIGateway chiama la `$default` route se l'espressione di selezione della rotta non può essere valutata rispetto al messaggio o se non viene trovata alcuna route corrispondente.

Espressioni di selezione dell'instradamento

Un'espressione di selezione dell'instradamento viene valutata quando il servizio seleziona l'instradamento da seguire per un messaggio in entrata. Il servizio utilizza la route il cui `routeKey` corrisponde esattamente al valore valutato. Se non trova alcuna corrispondenza ed è disponibile una route con una chiave di routing `$default`, verrà selezionata. Se nessuna route corrisponde al valore valutato e non è disponibile una route `$default`, il servizio restituisce un errore. Per WebSocket basedAPIs, l'espressione deve avere il seguente formato`$request.body.`*`{path_to_body_element}`*.

Ad esempio, supponiamo che tu stia inviando il seguente JSON messaggio:

```
{
    "service" : "chat",
    "action" : "join",
    "data" : {
        "room" : "room1234"
   }
}
```

Potresti voler selezionare il tuo API comportamento in base alla `action` proprietà. In questo caso, potresti definire la seguente espressione di selezione dell'instradamento:

```
$request.body.action
```

In questo esempio, `request.body` si riferisce al JSON payload del messaggio ed `.action` è un'JSONPathespressione. Puoi usare qualsiasi espressione di JSON percorso dopo`request.body`, ma tieni presente che il risultato verrà stringato. Ad esempio, se l'JSONPathespressione restituisce una matrice di due elementi, questo verrà presentato come stringa. `"[item1, item2]"` Per questo motivo, è opportuno che l'espressione restituisca un valore e non una matrice o un oggetto.

Puoi semplicemente utilizzare un valore statico oppure utilizzare più variabili. La tabella seguente mostra esempi con i relativi risultati valutati per il payload precedente.

Espressione	Risultato valutato	Descrizione
`$request.body.action`	`join`	Una variabile

Espressione	Risultato valutato	Descrizione
		da cui è stato rimosso il wrapping
`${request.body.action}`	`join`	Una variabile con wrapping
`${request.body.service}/${request.body.action}`	`chat/join`	Più variabili con valori statici
`${request.body.action}-${request.body.invalidPath}`	`join-`	Se non JSONPath viene trovato, la variabile viene risolta come «».
`action`	`action`	Valore statico
`\$default`	`$default`	Valore statico

Il risultato valutato viene utilizzato per trovare una route. Se esiste una route con una chiave di routing corrispondente, viene selezionato per elaborare il messaggio. Se non viene trovata alcuna rotta corrispondente, API Gateway tenta di trovare la `$default` rotta, se disponibile. Se il `$default` percorso non è definito, API Gateway restituisce un errore.

Imposta i percorsi per un WebSocket API in API Gateway

Quando ne crei uno nuovo per la prima volta WebSocket API, ci sono tre percorsi predefiniti: `$connect`$`disconnect`, e`$default`. È possibile crearli utilizzando la consoleAPI, o AWS CLI. Se lo desideri, puoi creare instradamenti personalizzati. Per ulteriori informazioni, consulta the section called “Panoramica di WebSocket APIs”.

> **Note**
>
> InCLI, puoi creare percorsi prima o dopo aver creato le integrazioni e puoi riutilizzare la stessa integrazione per più percorsi.

Crea un percorso utilizzando la console Gateway API

Per creare un percorso utilizzando la console API Gateway

1. Accedi alla console API Gateway, scegli Routes e scegli Routes. API
2. Seleziona Crea instradamento.
3. Per Chiave instradamento inserisci il nome della chiave instradamento. È possibile creare gli instradamenti predefiniti (`$connect`, `$disconnect` e `$default`) o un instradamento personalizzato.

   > **Note**
   >
   > Quando crei un instradamento personalizzato, non utilizzare il prefisso $ nel nome della chiave di instradamento. Questo prefisso è riservato per gli instradamenti predefiniti.

4. Seleziona e configura il tipo di integrazione per l'instradamento. Per ulteriori informazioni, consulta the section called “Imposta una richiesta di WebSocket API integrazione utilizzando la console API Gateway”.

Crea un percorso utilizzando il AWS CLI

Per creare un percorso utilizzando AWS CLI, chiama `create-route`come mostrato nell'esempio seguente:

```
aws apigatewayv2 --region us-east-1 create-route --api-id aabbccddee --route-key
 $default
```

Output di esempio:

```
{
    "ApiKeyRequired": false,
    "AuthorizationType": "NONE",
    "RouteKey": "$default",
    "RouteId": "1122334"
}
```

Specifica delle impostazioni per la richiesta di instradamento per **`$connect`**

Quando configuri il `$connect` percorso per il tuoAPI, sono disponibili le seguenti impostazioni opzionali per abilitare l'autorizzazione per il tuoAPI. Per ulteriori informazioni, consulta the section called “Instradamento `$connect`”.

- Authorization (Autorizzazione): se non è richiesta alcuna autorizzazione, puoi specificare `NONE`. In caso contrario, puoi specificare:
  - `AWS_IAM`per utilizzare AWS IAM politiche standard per controllare l'accesso al tuoAPI.
  - `CUSTOM`per implementare l'autorizzazione per an API specificando una funzione di autorizzazione Lambda creata in precedenza. L'autorizzatore può risiedere nel tuo AWS account o in un altro account. AWS Per ulteriori informazioni sui provider di autorizzazioni Lambda, consulta Usa gli API autorizzatori Gateway Lambda.

> **Note**
>
> Nella console API Gateway, l'`CUSTOM`impostazione è visibile solo dopo aver impostato una funzione di autorizzazione come descritto in. the section called “Configurare un autorizzatore Lambda (console)”

> **Important**
>
> L'impostazione di autorizzazione viene applicata all'intero percorsoAPI, non solo al `$connect` percorso. L'instradamento `$connect` protegge gli altri instradamenti, perché viene chiamato per ogni connessione.

- APIChiave richiesta: opzionalmente puoi richiedere una API chiave per il `$connect` percorso API di un utente. Puoi utilizzare API le chiavi insieme ai piani di utilizzo per controllare e tracciare l'accesso al tuoAPIs. Per ulteriori informazioni, consulta the section called “Piani di utilizzo”.

Configura la richiesta di **`$connect`** percorso utilizzando la console API Gateway

Per configurare la richiesta di `$connect` routing per un WebSocket API utilizzo della console API Gateway:

1. Accedi alla console API Gateway, scegli Routes e scegli Routes. API
2. In Instradamenti scegli `$connect` o crea un instradamento `$connect` seguendo the section called “Crea un percorso utilizzando la console Gateway API”.
3. Nella sezione Impostazioni della richiesta di instradamento scegli Modifica.
4. Per Autorizzazione seleziona un tipo di autorizzazione.
5. Per richiederne uno API per il `$connect` percorso, seleziona Richiedi API chiave.
6. Seleziona Salvataggio delle modifiche.

Imposta le risposte di routing per WebSocket APIs in API Gateway

WebSocket le rotte possono essere configurate per la comunicazione bidirezionale o unidirezionale. APIII gateway non trasmetterà la risposta del backend alla risposta di routing, a meno che non si imposti una risposta di routing.

> **Note**
>
> È possibile definire solo la risposta del `$default` percorso per WebSocket APIs. È possibile utilizzare una risposta di integrazione per manipolare la risposta da un servizio back-end. Per ulteriori informazioni, consulta the section called “Panoramica delle risposte di integrazione”.

È possibile configurare le risposte di routing e le espressioni di selezione della risposta utilizzando la console API Gateway AWS CLI o un AWS SDK.

Per ulteriori informazioni sulle espressioni di selezione delle risposte di instradamento, consulta the section called “”.

Argomenti

- Configurare una risposta di routing utilizzando la console API Gateway
- Imposta una risposta al percorso utilizzando il AWS CLI

Configurare una risposta di routing utilizzando la console API Gateway

Dopo aver creato WebSocket API e collegato una funzione proxy Lambda alla route predefinita, puoi configurare la risposta alla route utilizzando la console API Gateway:

1. Accedi alla console API Gateway, scegli un'integrazione della funzione Lambda WebSocket API con proxy sul `$default` percorso.
2. In Routes (Route), scegliere l'instradamento `$default`.
3. Scegli Abilita la comunicazione bidirezionale.
4. Seleziona Deploy (Implementa)API.
5. Distribuisci il tuo API su un palco.

Usa il seguente comando wscat per connetterti al tuo. API Per ulteriori informazioni su `wscat`, consulta the section called “Utilizzalo `wscat` per connetterti a un' WebSocket API e inviarle messaggi”.

```
wscat -c wss://api-id.execute-api.us-east-2.amazonaws.com/test
```

Premere il pulsante INVIO per chiamare l'instradamento predefinito. Il comando dovrebbe restituire il corpo della funzione Lambda.

Imposta una risposta al percorso utilizzando il AWS CLI

Per impostare una risposta al percorso per un WebSocket API utilizzando il AWS CLI, chiamate il `create-route-response`comando come mostrato nell'esempio seguente. È possibile identificare l'APIID e l'ID del percorso chiamando `get-apis`e `get-routes`.

```
aws apigatewayv2 create-route-response \
    --api-id aabbccddee \
    --route-id 1122334  \
    --route-response-key '$default'
```

Output di esempio:

```
{
    "RouteResponseId": "abcdef",
    "RouteResponseKey": "$default"
}
```

Configura una `$connect` route che richiede un WebSocket sottoprotocollo

I client possono utilizzare il `Sec-WebSocket-Protocol` campo per richiedere un WebSocket sottoprotocollo durante la connessione al tuo WebSocket API. Puoi configurare un'integrazione per il `$connect` percorso per consentire le connessioni solo se un client richiede un sottoprotocollo supportato dal tuoAPI.

La funzione Lambda di esempio seguente restituisce l'intestazione `Sec-WebSocket-Protocol` ai client. La funzione stabilisce una connessione con l'utente API solo se il client specifica il sottoprotocollo. `myprotocol`

Per un AWS CloudFormation modello che crea questo esempio API e l'integrazione del proxy Lambda, vedi. `ws-subprotocol.yaml`

```
export const handler = async (event) => {
    if (event.headers != undefined) {
        const headers = toLowerCaseProperties(event.headers);

        if (headers['sec-websocket-protocol'] != undefined) {
            const subprotocolHeader = headers['sec-websocket-protocol'];
            const subprotocols = subprotocolHeader.split(',');

            if (subprotocols.indexOf('myprotocol') >= 0) {
                const response = {
                    statusCode: 200,
                    headers: {
                        "Sec-WebSocket-Protocol" : "myprotocol"
                    }
```

```
                };
                return response;
            }
        }
    }

    const response = {
        statusCode: 400
    };

    return response;
};

function toLowerCaseProperties(obj) {
    var wrapper = {};
    for (var key in obj) {
        wrapper[key.toLowerCase()] = obj[key];
    }
    return wrapper;
}
```

Puoi utilizzarlo `wscat`per verificare che API consenta le connessioni solo se un client richiede un sottoprotocollo supportato da teAPI. I comandi seguenti utilizzano il flag `-s` per specificare i sottoprotocolli durante la connessione.

Il comando seguente tenta una connessione con un sottoprotocollo non supportato. Poiché il client ha specificato il sottoprotocollo `chat1`, l'integrazione Lambda restituisce un errore 400 e la connessione non riesce.

```
wscat -c wss://api-id.execute-api.region.amazonaws.com/beta -s chat1
error: Unexpected server response: 400
```

Il comando seguente include un sottoprotocollo supportato nella richiesta di connessione. L'integrazione Lambda consente la connessione.

```
wscat -c wss://api-id.execute-api.region.amazonaws.com/beta -s chat1,myprotocol
connected (press CTRL+C to quit)
```

Per ulteriori informazioni sull'invocazione WebSocket APIs, consulta. Richiama le API WebSocket

Controlla e gestisci l'accesso alle WebSocket API in API Gateway

API Gateway supporta diversi meccanismi per il controllo e la gestione dell'accesso all' WebSocket API.

Puoi usare i seguenti meccanismi per l'autenticazione e l'autorizzazione:

- I ruoli e le policy AWS IAM standard offrono controlli di accesso flessibili e robusti. Puoi utilizzare ruoli e policy IAM per controllare chi può creare e gestire le API e chi può richiamarle. Per ulteriori informazioni, consulta Controlla l'accesso alle WebSocket API con l'autorizzazione IAM.
- I tag IAM possono essere utilizzati insieme alle policy IAM per controllare l'accesso. Per ulteriori informazioni, consulta Utilizzo dei tag per controllare l'accesso alle risorse REST API di Gateway API.
- I provider di autorizzazioni Lambda sono funzioni Lambda che controllano l'accesso alle API. Per ulteriori informazioni, consulta Controlla l'accesso alle WebSocket API con AWS Lambda gli autorizzatori REQUEST.

Argomenti

- Controlla l'accesso alle WebSocket API con l'autorizzazione IAM
- Controlla l'accesso alle WebSocket API con AWS Lambda gli autorizzatori REQUEST

Controlla l'accesso alle WebSocket API con l'autorizzazione IAM

L'autorizzazione IAM nelle WebSocket API è simile a quella per le API REST, con le seguenti eccezioni:

- L'azione `execute-api` supporta `ManageConnections` oltre alle azioni esistenti (`Invoke`, `InvalidateCache`). `ManageConnections` controlla l'accesso all'API @connections.
- WebSocket i percorsi utilizzano un formato ARN diverso:

```
arn:aws:execute-api:region:account-id:api-id/stage-name/route-key
```

- L'API `@connections` usa lo stesso formato ARN delle API REST:

```
arn:aws:execute-api:region:account-id:api-id/stage-name/POST/@connections
```

> **Important**
>
> Quando usi l'autorizzazione IAM è necessario firmare le richieste con Signature Version 4 (SigV4).

Puoi, ad esempio, configurare la seguente policy per il client. Questo esempio consente a chiunque di inviare un messaggio (`Invoke`) per tutti gli instradamenti ad eccezione di un instradamento segreto nella fase `prod` e impedisce l'invio di un messaggio da qualunque utente ai client connessi (`ManageConnections`) in tutte le fasi.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:account-id:api-id/prod/*"
      ]
    },
    {
      "Effect": "Deny",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:account-id:api-id/prod/secret"
      ]
    },
    {
      "Effect": "Deny",
      "Action": [
        "execute-api:ManageConnections"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:account-id:api-id/*"
      ]
    }
  ]
```

```
}
```

Controlla l'accesso alle WebSocket API con AWS Lambda gli autorizzatori REQUEST

Una funzione di autorizzazione Lambda nelle WebSocket API è simile a quella delle API REST, con le seguenti eccezioni:

- È possibile usare solo una funzione di autorizzazione Lambda per l'instradamento `$connect`.
- Non è possibile utilizzare variabili di percorso (`event.pathParameters`), perché il percorso è fisso.
- `event.methodArn` è diverso dall'API REST equivalente, perché non ha un metodo HTTP. Nel caso di `$connect`, `methodArn` termina con `"$connect"`:

  ```
  arn:aws:execute-api:region:account-id:api-id/stage-name/$connect
  ```

- Le variabili di contesto in `event.requestContext` sono diverse da quelle per le API REST.

L'esempio seguente mostra un input a un autorizzatore per un'`REQUEST`API: WebSocket

```
{
    "type": "REQUEST",
    "methodArn": "arn:aws:execute-api:us-east-1:123456789012:abcdef123/default/
$connect",
    "headers": {
        "Connection": "upgrade",
        "content-length": "0",
        "HeaderAuth1": "headerValue1",
        "Host": "abcdef123.execute-api.us-east-1.amazonaws.com",
        "Sec-WebSocket-Extensions": "permessage-deflate; client_max_window_bits",
        "Sec-WebSocket-Key": "...",
        "Sec-WebSocket-Version": "13",
        "Upgrade": "websocket",
        "X-Amzn-Trace-Id": "...",
        "X-Forwarded-For": "...",
        "X-Forwarded-Port": "443",
        "X-Forwarded-Proto": "https"
    },
    "multiValueHeaders": {
        "Connection": [
            "upgrade"
        ],
```

```
        "content-length": [
            "0"
        ],
        "HeaderAuth1": [
            "headerValue1"
        ],
        "Host": [
            "abcdef123.execute-api.us-east-1.amazonaws.com"
        ],
        "Sec-WebSocket-Extensions": [
            "permessage-deflate; client_max_window_bits"
        ],
        "Sec-WebSocket-Key": [
            "..."
        ],
        "Sec-WebSocket-Version": [
            "13"
        ],
        "Upgrade": [
            "websocket"
        ],
        "X-Amzn-Trace-Id": [
            "..."
        ],
        "X-Forwarded-For": [
            "..."
        ],
        "X-Forwarded-Port": [
            "443"
        ],
        "X-Forwarded-Proto": [
            "https"
        ]
    },
    "queryStringParameters": {
        "QueryString1": "queryValue1"
    },
    "multiValueQueryStringParameters": {
        "QueryString1": [
            "queryValue1"
        ]
    },
    "stageVariables": {},
    "requestContext": {
```

```
        "routeKey": "$connect",
        "eventType": "CONNECT",
        "extendedRequestId": "...",
        "requestTime": "19/Jan/2023:21:13:26 +0000",
        "messageDirection": "IN",
        "stage": "default",
        "connectedAt": 1674162806344,
        "requestTimeEpoch": 1674162806345,
        "identity": {
            "sourceIp": "..."
        },
        "requestId": "...",
        "domainName": "abcdef123.execute-api.us-east-1.amazonaws.com",
        "connectionId": "...",
        "apiId": "abcdef123"
    }
}
```

Il seguente esempio di funzione di autorizzazione Lambda è una WebSocket versione della funzione di autorizzazione Lambda per le API REST in: the section called “Esempi aggiuntivi di funzioni di autorizzazione Lambda”

Node.js

```
  // A simple REQUEST authorizer example to demonstrate how to use request
  // parameters to allow or deny a request. In this example, a request is
  // authorized if the client-supplied HeaderAuth1 header and QueryString1 query
parameter
  // in the request context match the specified values of
  // of 'headerValue1' and 'queryValue1' respectively.
           export const handler = function(event, context, callback) {
   console.log('Received event:', JSON.stringify(event, null, 2));

  // Retrieve request parameters from the Lambda function input:
  var headers = event.headers;
  var queryStringParameters = event.queryStringParameters;
  var stageVariables = event.stageVariables;
  var requestContext = event.requestContext;

  // Parse the input for the parameter values
  var tmp = event.methodArn.split(':');
  var apiGatewayArnTmp = tmp[5].split('/');
  var awsAccountId = tmp[4];
```

```
   var region = tmp[3];
   var ApiId = apiGatewayArnTmp[0];
   var stage = apiGatewayArnTmp[1];
   var route = apiGatewayArnTmp[2];

   // Perform authorization to return the Allow policy for correct parameters and
   // the 'Unauthorized' error, otherwise.
   var authResponse = {};
   var condition = {};
    condition.IpAddress = {};

   if (headers.HeaderAuth1 === "headerValue1"
       && queryStringParameters.QueryString1 === "queryValue1") {
        callback(null, generateAllow('me', event.methodArn));
    }  else {
        callback("Unauthorized");
    }
}

// Helper function to generate an IAM policy
var generatePolicy = function(principalId, effect, resource) {
   // Required output:
   var authResponse = {};
    authResponse.principalId = principalId;
   if (effect && resource) {
       var policyDocument = {};
        policyDocument.Version = '2012-10-17'; // default version
       policyDocument.Statement = [];
       var statementOne = {};
        statementOne.Action = 'execute-api:Invoke'; // default action
       statementOne.Effect = effect;
        statementOne.Resource = resource;
        policyDocument.Statement[0] = statementOne;
        authResponse.policyDocument = policyDocument;
    }
   // Optional output with custom properties of the String, Number or Boolean type.
   authResponse.context = {
       "stringKey": "stringval",
       "numberKey": 123,
       "booleanKey": true
    };
   return authResponse;
}
```

```
var generateAllow = function(principalId, resource) {
   return generatePolicy(principalId, 'Allow', resource);
}

var generateDeny = function(principalId, resource) {
   return generatePolicy(principalId, 'Deny', resource);
}
```

Python

```
# A simple REQUEST authorizer example to demonstrate how to use request
# parameters to allow or deny a request. In this example, a request is
# authorized if the client-supplied HeaderAuth1 header and QueryString1 query
 parameter
# in the request context match the specified values of
# of 'headerValue1' and 'queryValue1' respectively.

import json


def lambda_handler(event, context):
    print(event)

    # Retrieve request parameters from the Lambda function input:
    headers = event['headers']
    queryStringParameters = event['queryStringParameters']
    stageVariables = event['stageVariables']
    requestContext = event['requestContext']

    # Parse the input for the parameter values
    tmp = event['methodArn'].split(':')
    apiGatewayArnTmp = tmp[5].split('/')
    awsAccountId = tmp[4]
    region = tmp[3]
    ApiId = apiGatewayArnTmp[0]
    stage = apiGatewayArnTmp[1]
    route = apiGatewayArnTmp[2]

    # Perform authorization to return the Allow policy for correct parameters
    # and the 'Unauthorized' error, otherwise.

    authResponse = {}
    condition = {}
```

```
    condition['IpAddress'] = {}

    if (headers['HeaderAuth1'] ==
            "headerValue1" and queryStringParameters["QueryString1"] ==
 "queryValue1"):
        response = generateAllow('me', event['methodArn'])
        print('authorized')
        return json.loads(response)
    else:
        print('unauthorized')
        return 'unauthorized'

    # Help function to generate IAM policy


def generatePolicy(principalId, effect, resource):
    authResponse = {}
    authResponse['principalId'] = principalId
    if (effect and resource):
        policyDocument = {}
        policyDocument['Version'] = '2012-10-17'
        policyDocument['Statement'] = []
        statementOne = {}
        statementOne['Action'] = 'execute-api:Invoke'
        statementOne['Effect'] = effect
        statementOne['Resource'] = resource
        policyDocument['Statement'] = [statementOne]
        authResponse['policyDocument'] = policyDocument

    authResponse['context'] = {
        "stringKey": "stringval",
        "numberKey": 123,
        "booleanKey": True
    }

    authResponse_JSON = json.dumps(authResponse)

    return authResponse_JSON


def generateAllow(principalId, resource):
    return generatePolicy(principalId, 'Allow', resource)
```

```
def generateDeny(principalId, resource):
    return generatePolicy(principalId, 'Deny', resource)
```

Per configurare la precedente funzione Lambda come funzione di autorizzazione per `REQUEST` WebSocket un'API, segui la stessa procedura utilizzata per le API REST.

Per configurare la route`$connect` per l'utilizzo di questo provider di autorizzazioni Lambda nella console, seleziona o crea la route `$connect`. Nella sezione Impostazioni della richiesta di instradamento scegli Modifica. Seleziona l'autorizzazione nel menu a discesa Autorizzazione, quindi scegli Salva modifiche.

Per verificare l'autorizzatore, è necessario creare una nuova connessione. La modifica dell'autorizzatore in `$connect` non influisce sul client già connesso. Quando ti connetti alla tua WebSocket API, devi fornire valori per tutte le fonti di identità configurate. Ad esempio, puoi connetterti inviando una stringa di query e un'intestazione valide utilizzando `wscat` come nell'esempio seguente:

```
wscat -c 'wss://myapi.execute-api.us-east-1.amazonaws.com/beta?
QueryString1=queryValue1' -H HeaderAuth1:headerValue1
```

Se tenti di connetterti senza un valore di identità valido, riceverai una risposta `401`:

```
wscat -c wss://myapi.execute-api.us-east-1.amazonaws.com/beta
error: Unexpected server response: 401
```

Integrazioni per WebSocket API in API Gateway

Dopo aver impostato un API percorso, devi integrarlo con un endpoint nel backend. Un endpoint di backend viene anche definito endpoint di integrazione e può essere una funzione Lambda, un HTTP endpoint o un'azione di servizio. AWS L'APIintegrazione ha una richiesta di integrazione e una risposta di integrazione.

In questa sezione, puoi imparare come configurare le richieste di integrazione e le risposte di integrazione per i tuoi WebSocket API.

Argomenti

- Imposta una richiesta di WebSocket API integrazione in API Gateway
- Imposta una risposta di WebSocket API integrazione in API Gateway

Imposta una richiesta di WebSocket API integrazione in API Gateway

La configurazione di una richiesta di integrazione comporta le seguenti operazioni:

- Selezione di una chiave di instradamento da integrare nel back-end.
- Specificazione dell'endpoint di backend da richiamare. WebSocket APIssupporta i seguenti tipi di integrazione:
  - `AWS_PROXY`
  - `AWS`
  - `HTTP_PROXY`
  - `HTTP`
  - `MOCK`

  Per ulteriori informazioni sui tipi di integrazione, vedere IntegrationTypein API Gateway V2 RESTAPI.
- Configurazione della modalità di trasformazione dei dati della richiesta di instradamento, se necessario, in dati di richiesta di integrazione specificando uno o più modelli di richiesta.

Imposta una richiesta di WebSocket API integrazione utilizzando la console API Gateway

Per aggiungere una richiesta di integrazione a un percorso WebSocket API utilizzando la console API Gateway

1. Accedi alla console API Gateway, scegli Routes e scegli Routes. API
2. In Routes (Route), scegliere la route.
3. Scegli la scheda Richiesta di integrazione, quindi seleziona Modifica nella sezione Impostazioni della richiesta di integrazione.
4. Per Tipo di integrazione scegli una delle seguenti opzioni:
   - Scegli la funzione Lambda solo se vuoi essere integrata con una AWS Lambda funzione che hai già creato in questo account o in un altro account. API

     Per creare una nuova funzione Lambda in AWS Lambda, impostare un'autorizzazione di risorsa sulla funzione Lambda o eseguire qualsiasi altra azione del servizio Lambda, scegli invece Service.AWS

- Scegli HTTPse vuoi API integrarlo con un endpoint esistente. HTTP Per ulteriori informazioni, consulta Integrazioni HTTP per le API REST in API Gateway.
- Scegli Mock se desideri generare API risposte direttamente da API Gateway, senza la necessità di un backend di integrazione. Per ulteriori informazioni, consulta Integrazioni fittizie per le API REST in API Gateway.
- Scegli il AWS servizio se desideri API integrarlo con un AWS servizio.
- Scegli il VPClink se API intendi utilizzare un `VpcLink` come endpoint di integrazione privato. Per ulteriori informazioni, consulta Integrazioni private per REST APIs in API Gateway.

5. Se scegli Funzione Lambda procedi come segue:

   a. Per Utilizza integrazione proxy Lambda seleziona la casella di controllo desiderata per utilizzare l'integrazione proxy Lambda o l'integrazione proxy Lambda tra più account.

   b. Per Funzione Lambda specifica la funzione in uno dei modi seguenti:

      - Se la funzione Lambda si trova nello stesso account, immetti il nome della funzione e quindi seleziona la funzione dall'elenco a discesa.

        > **Note**
        >
        > Il nome della funzione può facoltativamente includere l'alias o la specifica della versione, come in `HelloWorld`, `HelloWorld:1` o `HelloWorld:alpha`.

      - Se la funzione si trova in un account diverso, inserisci il ARN per la funzione.

   c. Per utilizzare il valore di timeout predefinito di 29 secondi, mantieni attiva l'opzione Timeout predefinito. Per impostare un timeout personalizzato, scegli Timeout predefinito e immetti un valore di timeout compreso tra `50` e `29000` millisecondi.

6. Se hai scelto HTTP, segui le istruzioni nel passaggio 4 dithe section called “ Configurazione di una richiesta di integrazione tramite la console”.
7. Se si sceglie Mock (Fittizio), andare alla fase Request Templates (Modelli di richiesta).
8. Se scegli Servizio AWS segui le istruzioni della fase 6 in the section called “ Configurazione di una richiesta di integrazione tramite la console”.
9. Se hai scelto il VPClink, procedi come segue:

   a. Per l'integrazione con VPC proxy, seleziona la casella di controllo se desideri che le tue richieste vengano inoltrate tramite proxy al tuo endpoint`VPCLink`.

b. Per il HTTPmetodo, scegli il tipo di HTTP metodo che più si avvicina al metodo nel backend. HTTP

c. Dall'elenco a discesa dei VPClink, seleziona un VPC link. È possibile selezionare `[Use Stage Variables]` e inserire **`${stageVariables.vpcLinkId}`** nella casella di testo sotto l'elenco.

   È possibile definire la variabile di `vpcLinkId` fase dopo averla distribuita API su uno stadio e impostarne il valore sull'ID di. `VpcLink`

d. Per Endpoint URL, inserisci il URL HTTP backend che desideri venga utilizzato da questa integrazione.

e. Per utilizzare il valore di timeout predefinito di 29 secondi, mantieni attiva l'opzione Timeout predefinito. Per impostare un timeout personalizzato, scegli Timeout predefinito e immetti un valore di timeout compreso tra `50` e `29000` millisecondi.

10. Scegli Save changes (Salva modifiche).

11. In Modelli di richiesta procedi come segue:

    a. Per immettere un'Espressione di selezione del modello scegli Modifica in Modelli di richiesta.

    b. Immetti un'Espressione di selezione del modello. Usa un'espressione che API Gateway cerca nel payload del messaggio. Se viene trovata, viene valutata e il risultato è un valore chiave del modello che viene utilizzato per selezionare il modello di mappatura dei dati che deve essere applicato ai dati nel payload del messaggio. Il modello di mappatura dei dati viene creato nel passaggio successivo. Scegli Modifica per salvare le modifiche.

    c. Scegli Crea modello per creare il modello di mappatura dei dati. Per Chiave del modello immetti il valore della chiave del modello utilizzato per selezionare il modello di mappatura dei dati da applicare ai dati nel payload del messaggio. Quindi, specifica un modello di mappatura. Scegli Crea modello.

       Per ulteriori informazioni sulle espressioni di selezione del modello, consulta the section called “Espressioni di selezione del modello”.

Imposta una richiesta di integrazione utilizzando AWS CLI

È possibile impostare una richiesta di integrazione per un percorso in a WebSocket API utilizzando AWS CLI come nell'esempio seguente, che crea un'integrazione fittizia:

1. Crea un file denominato `integration-params.json`, con i seguenti contenuti:

```
{"PassthroughBehavior": "WHEN_NO_MATCH", "TimeoutInMillis": 29000,
 "ConnectionType": "INTERNET", "RequestTemplates": {"application/json":
 "{\"statusCode\":200}"}, "IntegrationType": "MOCK"}
```

2. Eseguite il comando create-integration come illustrato nell'esempio seguente:

```
aws apigatewayv2 --region us-east-1 create-integration --api-id aabbccddee --cli-
input-json file://integration-params.json
```

Di seguito è riportato un output di esempio:

```
{
    "PassthroughBehavior": "WHEN_NO_MATCH",
    "TimeoutInMillis": 29000,
    "ConnectionType": "INTERNET",
    "IntegrationResponseSelectionExpression": "${response.statuscode}",
    "RequestTemplates": {
        "application/json": "{\"statusCode\":200}"
    },
    "IntegrationId": "0abcdef",
    "IntegrationType": "MOCK"
}
```

In alternativa, è possibile impostare una richiesta di integrazione per un'integrazione proxy utilizzando AWS CLI come illustrato nell'esempio seguente:

1. Creare una funzione Lambda nella console Lambda e assegnarle un ruolo di esecuzione Lambda di base.
2. Eseguite il comando create-integration come nell'esempio seguente:

```
aws apigatewayv2 create-integration --api-id aabbccddee --integration-type
 AWS_PROXY --integration-method POST --integration-uri arn:aws:apigateway:us-
east-1:lambda:path/2015-03-31/functions/arn:aws:lambda:us-
east-1:123412341234:function:simpleproxy-echo-e2e/invocations
```

Di seguito è riportato un output di esempio:

```
{
```

```
    "PassthroughBehavior": "WHEN_NO_MATCH",
    "IntegrationMethod": "POST",
    "TimeoutInMillis": 29000,
    "ConnectionType": "INTERNET",
    "IntegrationUri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123412341234:function:simpleproxy-echo-e2e/invocations",
    "IntegrationId": "abcdefg",
    "IntegrationType": "AWS_PROXY"
}
```

Formato di input di una funzione Lambda per l'integrazione del proxy per WebSocket APIs

Nell'integrazione del proxy Lambda, API Gateway mappa l'intera richiesta del client al `event` parametro di input della funzione Lambda di backend. L'esempio seguente mostra la struttura dell'evento di input dalla `$connect` route e dell'evento di input dalla `$disconnect` route che API Gateway invia a un'integrazione proxy Lambda.

Input from the $connect route

```
{
    headers: {
      Host: 'abcd123.execute-api.us-east-1.amazonaws.com',
      'Sec-WebSocket-Extensions': 'permessage-deflate; client_max_window_bits',
      'Sec-WebSocket-Key': '...',
      'Sec-WebSocket-Version': '13',
      'X-Amzn-Trace-Id': '...',
      'X-Forwarded-For': '192.0.2.1',
      'X-Forwarded-Port': '443',
      'X-Forwarded-Proto': 'https'
    },
    multiValueHeaders: {
      Host: [ 'abcd123.execute-api.us-east-1.amazonaws.com' ],
      'Sec-WebSocket-Extensions': [ 'permessage-deflate; client_max_window_bits' ],
      'Sec-WebSocket-Key': [ '...' ],
      'Sec-WebSocket-Version': [ '13' ],
      'X-Amzn-Trace-Id': [ '...' ],
      'X-Forwarded-For': [ '192.0.2.1' ],
      'X-Forwarded-Port': [ '443' ],
      'X-Forwarded-Proto': [ 'https' ]
    },
    requestContext: {
      routeKey: '$connect',
      eventType: 'CONNECT',
```

```
      extendedRequestId: 'ABCD1234=',
      requestTime: '09/Feb/2024:18:11:43 +0000',
      messageDirection: 'IN',
      stage: 'prod',
      connectedAt: 1707502303419,
      requestTimeEpoch: 1707502303420,
      identity: { sourceIp: '192.0.2.1' },
      requestId: 'ABCD1234=',
      domainName: 'abcd1234.execute-api.us-east-1.amazonaws.com',
      connectionId: 'AAAA1234=',
      apiId: 'abcd1234'
    },
    isBase64Encoded: false
  }
```

Input from the $disconnect route

```
{
    headers: {
      Host: 'abcd1234.execute-api.us-east-1.amazonaws.com',
      'x-api-key': '',
      'X-Forwarded-For': '',
      'x-restapi': ''
    },
    multiValueHeaders: {
      Host: [ 'abcd1234.execute-api.us-east-1.amazonaws.com' ],
      'x-api-key': [ '' ],
      'X-Forwarded-For': [ '' ],
      'x-restapi': [ '' ]
    },
    requestContext: {
      routeKey: '$disconnect',
      disconnectStatusCode: 1005,
      eventType: 'DISCONNECT',
      extendedRequestId: 'ABCD1234=',
      requestTime: '09/Feb/2024:18:23:28 +0000',
      messageDirection: 'IN',
      disconnectReason: 'Client-side close frame status not set',
      stage: 'prod',
      connectedAt: 1707503007396,
      requestTimeEpoch: 1707503008941,
      identity: { sourceIp: '192.0.2.1' },
```

```
      requestId: 'ABCD1234=',
      domainName: 'abcd1234.execute-api.us-east-1.amazonaws.com',
      connectionId: 'AAAA1234=',
      apiId: 'abcd1234'
    },
    isBase64Encoded: false
  }
```

Imposta una risposta di WebSocket API integrazione in API Gateway

La sezione seguente fornisce una breve panoramica delle risposte di integrazione per WebSocket API e su come impostare una risposta di integrazione per a WebSocket API.

Argomenti

- Panoramica delle risposte di integrazione
- Risposte di integrazione per la comunicazione bidirezionale
- Imposta una risposta di integrazione utilizzando la console API Gateway
- Imposta una risposta di integrazione utilizzando il AWS CLI

Panoramica delle risposte di integrazione

APILa risposta di integrazione di Gateway è un modo per modellare e manipolare la risposta di un servizio di backend. Esistono alcune differenze nella configurazione di una risposta REST API rispetto a quella di WebSocket API integrazione, ma concettualmente il comportamento è lo stesso.

WebSocket i percorsi possono essere configurati per la comunicazione bidirezionale o unidirezionale.

- Quando un percorso è configurato per la comunicazione bidirezionale, una risposta di integrazione consente di configurare le trasformazioni sul payload del messaggio restituito, in modo analogo alle risposte di integrazione per. REST APIs
- Se una route è configurata per la comunicazione unidirezionale, indipendentemente dalla configurazione della risposta di integrazione, non verrà restituita alcuna risposta sul WebSocket canale dopo l'elaborazione del messaggio.

APIIl gateway non passerà la risposta del backend alla risposta del percorso, a meno che non si imposti una risposta di routing. Per informazioni sull'impostazione di una risposta di instradamento, consultare the section called “Imposta le risposte di WebSocket API percorso”.

Risposte di integrazione per la comunicazione bidirezionale

Le integrazioni possono essere divise in integrazioni proxy e integrazioni non proxy.

> **Important**
>
> Per le integrazioni proxy, API Gateway trasmette automaticamente l'output del backend al chiamante come payload completo. Non esiste alcuna risposta di integrazione.

Per integrazioni non proxy, è necessario configurare almeno una risposta di integrazione:

- Idealmente, una delle risposte di integrazione deve agire come un metodo catch-all quando non è possibile effettuare una scelta esplicita. Questo caso predefinito viene rappresentato impostando una chiave di risposta di integrazione di `$default`.
- In tutti gli altri casi, la chiave di risposta di integrazione funziona come un'espressione regolare. Il formato adottato è `"/expression/"`.

Per le integrazioni non proxy: HTTP

- APIGateway tenterà di corrispondere al codice di HTTP stato della risposta del backend. In questo caso, la chiave di risposta di integrazione funzionerà come un'espressione regolare. Se non è possibile trovare una corrispondenza, viene scelta `$default` come risposta di integrazione.
- L'espressione di selezione del modello, come descritto in precedenza, funziona in modo identico. Ad esempio:
  - `/2\d\d/`: ricevi e trasforma risposte andate a buon fine
  - `/4\d\d/`: ricevi e trasforma errori di richiesta non valida
  - `$default`: ricevi e trasforma tutte le risposte impreviste

Per ulteriori informazioni sulle espressioni di selezione del modello, consultare the section called “Espressioni di selezione del modello”.

Imposta una risposta di integrazione utilizzando la console API Gateway

Per impostare una risposta di integrazione del percorso per un WebSocket API utilizzo della console API Gateway:

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli il tuo WebSocket API e scegli il tuo percorso.
3. Scegli la scheda Richiesta di integrazione, quindi seleziona Crea risposta di integrazione nella sezione Impostazioni della risposta di integrazione.
4. Per Chiave della risposta inserisci un valore che verrà individuato nella chiave della risposta del messaggio in uscita dopo aver valutato l'espressione di selezione di risposta. Ad esempio, puoi inserire **/4\d\d/** per ricevere e trasformare gli errori di richiesta non valida o **$default** per ricevere e trasformare tutte le risposte che corrispondono all'espressione di selezione del modello.
5. Per Espressione di selezione del modello immetti un'espressione di selezione per valutare il messaggio in uscita.
6. Scegli Crea risposta.
7. È inoltre possibile definire un modello di mappatura per configurare le trasformazioni del payload dei messaggi restituiti. Scegli Crea modello.
8. Immettere un nome per la chiave. Se è stata scelta l'espressione di selezione del modello predefinita, immettere**\$default**.
9. Per Modello della risposta immetti il modello di mappatura nell'editor di codice.
10. Scegli Crea modello.
11. Scegli Deploy API per implementare il tuo. API

Usa il seguente comando wscat per connetterti al tuo. API Per ulteriori informazioni su `wscat`, consulta the section called “Utilizzalo `wscat` per connetterti a un' WebSocket API e inviarle messaggi”.

```
wscat -c wss://api-id.execute-api.us-east-2.amazonaws.com/test
```

Quando viene richiamato l'instradamento, dovrebbe venire restituito il payload del messaggio restituito.

Imposta una risposta di integrazione utilizzando il AWS CLI

Per impostare una risposta di integrazione per un WebSocket API utilizzando il comando AWS CLI chiama il create-integration-responsecomando. Il CLI comando seguente mostra un esempio di creazione di una risposta di `$default` integrazione:

```
aws apigatewayv2 create-integration-response \
    --api-id vaz7da96z6 \
    --integration-id a1b2c3 \
    --integration-response-key '$default'
```

Richiedi la convalida delle API in WebSocket API Gateway

Puoi configurare API Gateway per eseguire la convalida su una route prima di passare alla richiesta di integrazione. Se la convalida fallisce, API Gateway fallisce la richiesta senza chiamare il backend, invia una risposta gateway «Bad request body» al client e pubblica i risultati della convalida in Logs. CloudWatch L'utilizzo della convalida in questo modo riduce le chiamate non necessarie al back-end dell'API.

Espressioni di selezione del modello

Puoi utilizzare un'espressione di selezione del modello per convalidare dinamicamente le richieste all'interno della stessa route. La convalida del modello si verifica se si fornisce un'espressione di selezione del modello per integrazioni proxy o non proxy. Potrebbe essere necessario definire il modello `$default` come un fallback quando non viene trovato alcun modello corrispondente. Se non esiste un modello corrispondente e `$default` non è definito, la convalida non va a buon fine. L'aspetto dell'espressione di selezione è simile a `Route.ModelSelectionExpression` e valuta la chiave per `Route.RequestModels`.

Quando definisci un percorso per un' WebSocket API, puoi facoltativamente specificare un'espressione di selezione del modello. Questa espressione viene valutata per selezionare il modello da utilizzare per la convalida del corpo quando si riceve una richiesta. L'espressione restituisce una delle voci presenti nel di una route `requestmodels`.

Un modello viene espresso come schema JSON e descrive la struttura dati del corpo della richiesta. La natura di queste espressioni di selezione consente di scegliere in modo dinamico il modello in base al quale eseguire la convalida in fase di runtime per una determinata route. Per informazioni su come creare un modello, consulta the section called “Modelli di dati per REST APIs”.

Configurazione della convalida delle richieste tramite la console Gateway Amazon API

L'esempio seguente mostra come impostare la convalida delle richieste su un percorso.

Innanzitutto, crei un modello e poi crei un percorso. Successivamente, configuri la convalida della richiesta sulla rotta appena creata. Infine, implementate e testate la vostra API. Per completare questo tutorial, hai bisogno di un' WebSocket API `$request.body.action` come espressione di selezione del percorso e di un endpoint di integrazione per la tua nuova rotta.

Per la connessione all'API è inoltre necessario `wscat`. Per ulteriori informazioni, consulta the section called “Utilizzalo `wscat` per connetterti a un' WebSocket API e inviarle messaggi”.

Creazione di un modello

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli un' WebSocket API.
3. Nel riquadro di navigazione principale seleziona Modelli.
4. Scegli Crea modello.
5. Per Nome, immetti **`emailModel`**.
6. Per Tipo di contenuto inserisci **`application/json`**.
7. Per Schema modello immetti il seguente modello:

   ```
   {
       "$schema": "http://json-schema.org/draft-04/schema#",
       "type" : "object",
       "required" : [ "address"],
       "properties" : {
           "address": {
               "type": "string"
           }
       }
   }
   ```

   Questo modello richiede che la richiesta contenga un indirizzo email.
8. Selezionare Salva.

In questo passaggio, crei un percorso per la tua WebSocket API.

Per creare una route

1. Nel pannello di navigazione principale, scegli Percorsi.
2. Selezionare Create Route (Crea route).
3. Per Route key (Chiave routing), inserire **`sendMessage`**.
4. Scegli un tipo di integrazione e specifica un endpoint di integrazione. Per ulteriori informazioni, consulta the section called “Integrazioni”.
5. Selezionare Create Route (Crea route).

In questo passaggio, si imposta la convalida della richiesta per il `sendMessage` percorso.

Per impostare la convalida della richiesta

1. Nella scheda Richiesta di percorso, in Impostazioni di richiesta di percorso, scegli Modifica.
2. Per Espressione di selezione del modello, immettete**`${request.body.messageType}`**.

   API Gateway utilizza la `messageType` proprietà per convalidare la richiesta in arrivo.
3. Scegli Aggiungi modello di richiesta.
4. Per Model key, inserisci**`email`**.
5. Per Modello, scegli EmailModel.

   API Gateway convalida i messaggi in arrivo con la `messageType` proprietà impostata su `email` rispetto a questo modello.

   > **Note**
   >
   > Se API Gateway non è in grado di abbinare l'espressione di selezione del modello a una chiave del modello, seleziona il `$default` modello. Se non esiste un `$default` modello, la convalida fallisce. Per le API di produzione, ti consigliamo di creare un `$default` modello.

6. Seleziona Salvataggio delle modifiche.

In questa fase, distribuisci e testerai la tua API.

Per distribuire e testare la tua API

1. Seleziona Deploy API (Distribuisci API).
2. Scegliere la fase desiderata dall'elenco a discesa oppure immettere il nome di una nuova fase.
3. Seleziona Deploy (Implementa).
4. Nel riquadro di navigazione principale scegli Fasi.
5. Copia l' WebSocket URL della tua API. L'URL dovrebbe essere del tipo `wss://abcdef123.execute-api.us-east-2.amazonaws.com/production`.
6. Apri un nuovo terminale ed esegui il wscat comando con i seguenti parametri.

```
wscat -c wss://abcdef123.execute-api.us-west-2.amazonaws.com/production
```

```
Connected (press CTRL+C to quit)
```

7. Usa il seguente comando per testare la tua API.

```
{"action": "sendMessage", "messageType": "email"}
```

```
{"message": "Invalid request body", "connectionId":"ABCD1=234",
 "requestId":"EFGH="}
```

API Gateway non riuscirà a completare la richiesta.

Usa il comando successivo per inviare una richiesta valida alla tua API.

```
{"action": "sendMessage", "messageType": "email", "address":
 "mary_major@example.com"}
```

Trasformazioni dei dati per le API in WebSocket API Gateway

In API Gateway, la richiesta del metodo di un' WebSocket API può accettare un payload in un formato diverso dal payload della richiesta di integrazione corrispondente, come richiesto nel backend. Analogamente, il back-end potrebbe restituire un payload delle risposta di integrazione diverso dal payload della risposta di metodo, in base alle aspettative del front-end.

API Gateway consente di utilizzare i modelli di mappatura per mappare il payload da una richiesta del metodo alla richiesta di integrazione corrispondente e da una risposta di integrazione alla corrispondente risposta del metodo. Specifica un'espressione di selezione del modello per determinare quale modello utilizzare per eseguire le trasformazioni dei dati necessarie.

È possibile utilizzare le mappature dei dati per mappare i dati da una richiesta di instradamento a un'integrazione back-end. Per ulteriori informazioni, consulta the section called “Mappatura dei dati”.

Modelli di mappatura e modelli

Un modello di mappatura è uno script espresso in Velocity Template Language (VTL) e applicato al payload tramite Espressioni JSONPath. Per ulteriori informazioni sui modelli di mappatura API Gateway, consulta Modelli di mappatura per REST APIs.

Il payload può avere un modello di dati in base alla bozza 4 dello schema JSON. Per generare un modello di mappatura, non è necessario definire un modello. Tuttavia, un modello può essere utile per crearne un altro, poiché API Gateway genera un piano di modello in base a un modello fornito. Per ulteriori informazioni sui modelli API Gateway, consulta Modelli di dati per REST APIs.

Espressioni di selezione del modello

Per trasformare un payload con un modello di mappatura, si specifica un'espressione di selezione del modello WebSocket API in una richiesta di integrazione o in una risposta di integrazione. Questa espressione viene valutata per determinare il modello di input o di output (se disponibili) da usare per trasformare il corpo della richiesta nel corpo della richiesta di integrazione (tramite un modello di input) o il corpo della risposta di integrazione nel corpo della risposta di instradamento (tramite un modello di output).

`Integration.TemplateSelectionExpression` supporta `${request.body.jsonPath}` e valori statici.

`IntegrationResponse.TemplateSelectionExpression` supporta `${request.body.jsonPath}`, `${integration.response.statuscode}`, `${integration.response.header.headerName}`, `${integration.response.multivalueheader.headerName}` e valori statici.

Espressioni di selezione della risposta di integrazione

Quando imposti una risposta di integrazione per un' WebSocket API, puoi facoltativamente specificare un'espressione di selezione della risposta di integrazione. Questa espressione determina

quale `IntegrationResponse` deve essere selezionata quando viene restituita un'integrazione. Il valore di questa espressione è attualmente limitato da API Gateway, come definito di seguito. Tieni presente che questa espressione è rilevante solo per le integrazioni non proxy; un'integrazione proxy restituisce semplicemente il payload della risposta all'intermediario senza alcuna modellazione o modifica.

A differenza di altre espressioni di selezione precedenti, questa espressione supporta attualmente un formato di corrispondenza di modelli. L'espressione deve essere racchiusa tra barre.

Attualmente il valore è fisso e dipende dal valore di `integrationType`:

- Per le integrazioni basate su Lambda, è `$integration.response.body.errorMessage`.
- Per le integrazioni `HTTP` e `MOCK`, è `$integration.response.statuscode`.
- Per `HTTP_PROXY` e `AWS_PROXY`, l'espressione non viene utilizzata poiché si richiede che il payload raggiunga il chiamante.

Configurare la mappatura dei dati per WebSocket APIs in API Gateway

La mappatura dei dati consente di mappare i dati da una richiesta di instradamento a un'integrazione back-end.

> **Note**
>
> La mappatura dei dati per WebSocket APIs non è supportata in. AWS Management ConsoleÈ necessario utilizzare AWS CLI AWS CloudFormation, o an SDK per configurare la mappatura dei dati.

Argomenti

- Come mappare i dati di richiesta di instradamento ai parametri di richiesta di integrazione
- Esempi

Come mappare i dati di richiesta di instradamento ai parametri di richiesta di integrazione

I parametri della richiesta di integrazione possono essere mappati da tutti i parametri di richiesta di instradamento definiti, dal corpo della richiesta `context o` dalle variabili `stage` e dai valori statici.

La tabella seguente mostra le espressioni di mappatura dei dati delle richieste di integrazione. Nella tabella, *PARAM_NAME* è il nome di un parametro di richiesta di percorso del tipo di parametro specificato. Deve corrispondere all'espressione regolare `'^[a-zA-Z0-9._$-]+$]'`. *JSONPath_EXPRESSION* è un'JSONPathespressione per un JSON campo del corpo della richiesta.

Espressioni di mappatura dei dati di richieste di integrazione

Origine dati mappata	Espressione di mappatura
Stringa di query di richiesta (supportata solo per l'instradamento `$connect`)	`route.request.querystring.` *PARAM_NAME*
Intestazione della richiesta (supportata solo per l'instradamento `$connect`)	`route.request.header.` *PARAM_NAME*
Stringa di query della richiesta a più valori (supportata solo per l'instradamento `$connect`)	`route.request.multivaluequerystring.` *PARAM_NAME*
Intestazione della richiesta a più valori (supportata solo per l'instradamento `$connect`)	`route.request.multivalueheader.` *PARAM_NAME*
Corpo di richiesta	`route.request.body.` *JSONPath_EXPRESSION*
Variabili di fase	`stageVariables.` *VARIABLE_NAME*
Variabili di contesto	`context.`*VARIABLE_NAME* che deve essere una delle variabili di contesto supportate.
Valore statico	*'STATIC_VALUE'* . La *STATIC_VALUE* è una stringa letterale e deve essere racchiusa tra virgolette singole.

Esempi

I seguenti AWS CLI esempi configurano le mappature dei dati. Per un AWS CloudFormation modello di esempio, vedere. `websocket-data-mapping.yaml`

Mappa quello di un client connectionId a un'intestazione in una richiesta di integrazione

Il comando di esempio seguente associa un `connectionId` di un client a un'intestazione `connectionId` nella richiesta a un'integrazione back-end.

```
aws apigatewayv2 update-integration \
    --integration-id abc123 \
    --api-id a1b2c3d4 \
    --request-parameters
 'integration.request.header.connectionId'='context.connectionId'
```

Mappatura di un parametro di stringa di query a un'intestazione in una richiesta di integrazione

I comandi di esempio seguenti mappano un parametro stringa di query `authToken` a un'intestazione `authToken` nella richiesta di integrazione.

Innanzitutto, aggiungere il parametro della stringa di query `authToken` ai parametri di richiesta dell'instradamento.

```
aws apigatewayv2 update-route --route-id 0abcdef \
    --api-id a1b2c3d4 \
    --request-parameters '{"route.request.querystring.authToken": {"Required": false}}'
```

Quindi, mappare il parametro della stringa di query all'intestazione `authToken` nella richiesta di integrazione back-end.

```
aws apigatewayv2 update-integration \
    --integration-id abc123 \
    --api-id a1b2c3d4 \
    --request-parameters
 'integration.request.header.authToken'='route.request.querystring.authToken'
```

Se necessario, eliminare il parametro della stringa di query `authToken` ai parametri di richiesta dell'instradamento.

```
aws apigatewayv2 delete-route-request-parameter \
    --route-id 0abcdef \
    --api-id a1b2c3d4 \
    --request-parameter-key 'route.request.querystring.authToken'
```

WebSocket APIriferimento al modello di mappatura per API Gateway

Questa sezione riassume l'insieme di variabili attualmente supportate WebSocket APIs in API Gateway.

Parametro	Descrizione
`$context.connectionId`	Un ID univoco per la connessione, che può essere utilizzato per effettuare un callback al client.
`$context.connectedAt`	L'ora della connessione in formato Epoch.
`$context.domainName`	Un nome di dominio per. WebSocket API Può essere utilizzato per effettuare un callback al client (invece di un valore hardcoded).
`$context.eventType`	Il tipo di evento: `CONNECT`, `MESSAGE` o `DISCONNECT` .
`$context.messageId`	Un ID univoco sul lato server per un messaggio . Disponibile solo quando `$context.eventType` è `MESSAGE`.
`$context.routeKey`	La chiave di instradamento selezionata.
`$context.requestId`	Come `$context.extendedRequestId` .
`$context.extendedRequestId`	Un ID generato automaticamente per la API chiamata, che contiene altre informazioni utili per il debugg/risoluzione dei problemi.
`$context.apiId`	L'identificatore API che Gateway assegna al tuo. API
`$context.authorizer.principalId`	L'identificazione dell'utente principale associata al token inviato dal client e restituito da una funzione Lambda di autorizzazione API

Parametro	Descrizione
	Gateway Lambda (precedentemente nota come autorizzazione personalizzata).
`$context.authorizer.` *`property`*	Il valore stringato della coppia chiave-valore specificata della `context` mappa restituito da una funzione di autorizzazione Gateway API Lambda. Ad esempio, se le autorizzazioni restituiscono la mappa `context` seguente: `"context" : { "key": "value", "numKey": 1, "boolKey": true }` la chiamata di `$context.authorizer.key` restituisce la stringa `"value"`, la chiamata di `$context.authorizer.numKey` restituisce la stringa `"1"` e la chiamata di `$context.authorizer.boolKey` restituisce la stringa `"true"`.
`$context.error.messageString`	Valore `$context.error.message` tra virgolette, ovvero `"$context.error.message"` .
`$context.error.validationErrorString`	Stringa contenente un messaggio dettagliato di errore di convalida.
`$context.identity.accountId`	L'ID dell'account associato alla richiesta. AWS
`$context.identity.apiKey`	La chiave API del proprietario associata alla richiesta con chiave abilitataAPI.
`$context.identity.apiKeyId`	L'ID della API chiave associato alla richiesta abilitata alla chiave API

Parametro	Descrizione
`$context.identity.caller`	Identificatore dell'entità principale del chiamante da cui proviene la richiesta.
`$context.identity.cognitoAuthenticationProvider`	Un elenco separato da virgole di tutti i provider di autenticazione Amazon Cognito utilizzati dal chiamante che effettua la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito. Ad esempio, per un'identità di un pool di utenti Amazon Cognito, `cognito-idp.`*`region`*`.amazonaws.com/`*`user_pool_id`*`,cognito-idp.`*`region`*`.amazonaws.com/`*`user_pool_id`*`:CognitoSignIn:`*`token subject claim`* Per informazioni sui provider di autenticazione di Amazon Cognito disponibili, consulta Using Federated Identities nella Amazon Cognito Developer Guide.
`$context.identity.cognitoAuthenticationType`	Tipo di autenticazione Amazon Cognito dell'intermediario da cui proviene la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito. I valori possibili includono `authenticated` per le identità autenticate e `unauthenticated` per le identità non autenticate.
`$context.identity.cognitoIdentityId`	ID identità di Amazon Cognito dell'intermediario da cui proviene la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito.

Parametro	Descrizione
`$context.identity.cognitoIdentityPoolId`	ID pool di identità di Amazon Cognito dell'intermediario da cui proviene la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito.
`$context.identity.sourceIp`	L'indirizzo IP di origine della TCP connessione immediata che effettua la richiesta all'endpoint Gateway. API
`$context.identity.user`	Identificatore dell'entità principale dell'utente da cui proviene la richiesta.
`$context.identity.userAgent`	L'agente utente del API chiamante.
`$context.identity.userArn`	L'Amazon Resource Name (ARN) dell'utente effettivo identificato dopo l'autenticazione.
`$context.requestTime`	L'ora della richiesta CLFin formato -formattato ()`dd/MMM/yyyy:HH:mm:ss +-hhmm` .
`$context.requestTimeEpoch`	L'ora della richiesta in formato epoca (Unix epoch) in millisecondi.
`$context.stage`	La fase di distribuzione della API chiamata (ad esempio, Beta o Prod).
`$context.status`	Lo stato della risposta.
`$input.body`	Restituisce il payload non elaborato come stringa.

Parametro	Descrizione
`$input.json(x)`	Questa funzione valuta un'JSONPathespressione e restituisce i risultati sotto forma di stringaJSON. Ad esempio, `$input.json('$.pets')` restituirà una JSON stringa che rappresenta la struttura degli animali domestici. Per ulteriori informazioni suJSONPath, vedere JSONPathor JSONPathfor Java.

Parametro	Descrizione
`$input.path(x)`	Accetta una stringa di JSONPath espressio ne (x) e restituisce una rappresentazione JSON oggettuale del risultato. Ciò consente di accedere e manipolare gli elementi del payload in modo nativo in Apache Velocity Template Language (). VTL Ad esempio, se l'espressione `$input.path('$.pets')` restituisce un oggetto in questo modo: `[ { "id": 1, "type": "dog", "price": 249.99 }, { "id": 2, "type": "cat", "price": 124.99 }, { "id": 3, "type": "fish", "price": 0.99 } ]` `$input.path('$.pets').count()` restituisce `"3"`. Per ulteriori informazioni suJSONPath, vedere or for Java. JSONPathJSONPath
`$stageVariables.` *`<variable_name>`*	*`<variable_name>`* rappresenta il nome di una variabile di fase.

Parametro	Descrizione
`$stageVariables['<variable_name>']`	`<variable_name>` rappresenta qualsiasi nome di variabile di fase.
`${stageVariables['<variable_name>']}`	`<variable_name>` rappresenta qualsiasi nome di variabile di fase.
`$util.escapeJavaScript()`	Sfugge ai caratteri di una stringa utilizzando le regole delle JavaScript stringhe. **Note** Questa funzione trasforma qualsiasi virgoletta singola (`'`) in virgoletta preceduta da un carattere escape (`\'`). Tuttavia, le virgolette singole con escape non sono valide in. JSON Pertanto, quando l'output di questa funzione viene utilizzato in una JSON proprietà, è necessario riconvertire le virgolette singole con escape (`\'`) in virgolette singole regolari (`'`). Questo viene mostrato nell'esempio seguente: `$util.escapeJavaScript( data).replaceAll("\\'","'")`

Parametro	Descrizione

`$util.parseJson()`	Prende «stringified» JSON e restituisce una rappresentazione oggettuale del risultato. È possibile utilizzare il risultato di questa funzione per accedere e manipolare gli elementi del payload in modo nativo in Apache Velocity Template Language (). VTL Ad esempio, in presenza del payload seguente:
	`{"errorMessage":"{\"key1\":\"var1\", \"key2\":{\"arr\":[1,2,3]}}"}`
	E se usi il modello di mappatura seguente:
	`#set ($errorMessageObj = $util.parseJson($input.path('$.errorMessage')))`
	`{`
	`"errorMessageObjKey2ArrVal" : $errorMessageObj.key2.arr[0]`
	`}`
	Otterrai l'output seguente:
	`{`
	`"errorMessageObjKey2ArrVal" : 1`
	`}`

`$util.urlEncode()`	Converte una stringa nel formato «application/». x-www-form-urlencoded
`$util.urlDecode()`	Decodifica una stringa «application/». x-www-form-urlencoded
`$util.base64Encode()`	Codifica i dati in una stringa con codifica base64.

Parametro	Descrizione
`$util.base64Decode()`	Decodifica i dati da una stringa con codifica base64.

Tipi di supporti binari per le WebSocket API in API Gateway

Le API WebSocket API Gateway attualmente non supportano i frame binari nei payload dei messaggi in arrivo. Se un'app client invia un frame binario, viene rifiutato da API Gateway e la connessione al client viene annullata con codice 1003.

Esiste una soluzione alternativa a questo comportamento. Se il client invia dati binari con codifica di testo (ad esempio, base64) come un frame di testo, puoi impostare la proprietà `contentHandlingStrategy` dell'integrazione su `CONVERT_TO_BINARY` per convertire il payload da stringa con codifica base64 in binario.

Per restituire una risposta di instradamento per un payload binario in integrazioni non proxy, puoi impostare la proprietà `contentHandlingStrategy` della risposta di integrazione su `CONVERT_TO_TEXT` per convertire il payload da binario in stringa con codifica base64.

Richiama le API WebSocket

Dopo aver distribuito l' WebSocket API, le applicazioni client possono connettersi ad essa e inviarle messaggi, mentre il servizio di backend può inviare messaggi alle applicazioni client connesse:

- Puoi utilizzarla `wscat` per connetterti alla tua WebSocket API e inviarle messaggi per simulare il comportamento del client. Consulta the section called “Utilizzalo `wscat` per connetterti a un' WebSocket API e inviarle messaggi”.
- Puoi utilizzare l'API @connections dal servizio di back-end per inviare un messaggio di callback a un client connesso, ottenere informazioni di connessione o scollegare il client. Consulta the section called “Utilizzo di comandi `@connections` nel servizio di back-end”.
- Un'applicazione client può utilizzare la propria WebSocket libreria per richiamare l' WebSocketAPI.

Utilizzalo **`wscat`** per connetterti a un' WebSocket API e inviarle messaggi

L'`wscat`utilità è uno strumento utile per testare un' WebSocket API creata e distribuita in API Gateway. Puoi installare e utilizzare `wscat` nel modo seguente:

1. Scarica `wscat` dall'indirizzo https://www.npmjs.com/package/wscat.
2. Installa `wscat` eseguendo i comandi seguenti.

```
npm install -g wscat
```

3. Per connetterti all'API, esegui il comando `wscat` come mostrato nell'esempio seguente. Questo esempio presuppone che l'impostazione `Authorization` sia `NONE`.

```
wscat -c wss://aabbccddee.execute-api.us-east-1.amazonaws.com/test/
```

   Sarà necessario sostituire *`aabbccddee`* con l'ID API effettivo, visualizzato nella console API Gateway o restituito dal comando `create-api` della AWS CLI .

   Inoltre, se l'API si trova in una regione diversa da `us-east-1`, sarà necessario sostituire la regione corretta.

4. Per testare l'API, immetti un messaggio, ad esempio il seguente mentre sei connesso:

```
{"{jsonpath-expression}":"{route-key}"}
```

   in cui *`{jsonpath-expression}`* è un'espressione JSONPath e *`{route-key}`* è una chiave route per l'API. Ad esempio:

```
{"action":"action1"}
{"message":"test response body"}
```

   Per ulteriori informazioni su JSONPath, consulta la pagina relativa a JSONPath o JSONPath per Java.

5. Per disconnetterti dall'API, immetti `ctrl-C`.

Utilizzo di comandi **`@connections`** nel servizio di back-end

Il servizio di backend può utilizzare le seguenti richieste HTTP di WebSocket connessione per inviare un messaggio di callback a un client connesso, ottenere informazioni sulla connessione o disconnettere il client.

> **Important**
>
> Queste richieste utilizzano l'autorizzazione IAM, pertanto occorre firmarle con Signature Version 4 (SigV4). Per eseguire questa operazione, puoi utilizzare l'API di gestione di API Gateway. Per ulteriori informazioni, consulta. ApiGatewayManagementApi

Nel comando seguente, è necessario sostituirlo *{api-id}* con l'ID API effettivo, che viene visualizzato nella console API Gateway o restituito dal comando AWS CLI create-api. È necessario stabilire la connessione prima di utilizzare questo comando.

Per inviare un messaggio di callback al client, utilizza:

```
POST https://{api-id}.execute-api.us-east-1.amazonaws.com/{stage}/
@connections/{connection_id}
```

Puoi testare questa richiesta utilizzando `Postman` o chiamando `awscurl` come nell'esempio seguente:

```
awscurl --service execute-api -X POST -d "hello world" https://{prefix}.execute-api.us-
east-1.amazonaws.com/{stage}/@connections/{connection_id}
```

Il comando deve essere codificato tramite URL come nell'esempio seguente:

```
awscurl --service execute-api -X POST -d "hello world" https://aabbccddee.execute-
api.us-east-1.amazonaws.com/prod/%40connections/R0oXAdfD0kwCH6w%3D
```

Per ottenere l'ultimo stato di connessione del client, utilizza:

```
GET https://{api-id}.execute-api.us-east-1.amazonaws.com/{stage}/
@connections/{connection_id}
```

Per scollegare il client, utilizza:

```
DELETE https://{api-id}.execute-api.us-east-1.amazonaws.com/{stage}/
@connections/{connection_id}
```

Puoi creare dinamicamente un URL di callback utilizzando le variabili `$context` nell'integrazione. Ad esempio, se si utilizza un'integrazione proxy Lambda con una funzione Lambda `Node.js`, puoi creare l'URL e inviare un messaggio a un client connesso come segue:

```
import {
  ApiGatewayManagementApiClient,
  PostToConnectionCommand,
} from "@aws-sdk/client-apigatewaymanagementapi";

export const handler = async (event) => {
  const domain = event.requestContext.domainName;
  const stage = event.requestContext.stage;
  const connectionId = event.requestContext.connectionId;
  const callbackUrl = `https://${domain}/${stage}`;
  const client = new ApiGatewayManagementApiClient({ endpoint: callbackUrl });

  const requestParams = {
    ConnectionId: connectionId,
    Data: "Hello!",
  };

  const command = new PostToConnectionCommand(requestParams);

  try {
    await client.send(command);
  } catch (error) {
    console.log(error);
  }

  return {
    statusCode: 200,
  };
};
```

Quando si invia un messaggio di callback, la funzione Lambda deve disporre dell'autorizzazione per chiamare l'API di gestione di Gateway API. Se pubblichi un messaggio prima che venga stabilita la connessione o dopo la disconnessione del client, potresti ricevere un errore contenente `GoneException`.

Pubblica WebSocket API che i clienti possono richiamare

La semplice creazione e sviluppo di un'API di API Gateway non la rende automaticamente richiamabile dagli utenti. Per renderla richiamabile, è necessario distribuire l'API in una fase. Inoltre, potrebbe essere necessario personalizzare l'URL che verrà utilizzato dagli utenti per accedere all'API. Puoi assegnarli un dominio che sia coerente con il tuo marchio o che sia più facile da ricordare dell'URL predefinito dell'API.

In questa sezione viene descritto come distribuire l'API e personalizzare l'URL fornito agli utenti per accedervi.

> **Note**
>
> Per aumentare la sicurezza delle API API Gateway, il dominio `execute-api.{region}.amazonaws.com`è registrato nella Public Suffix List (PSL). Per una maggiore sicurezza, consigliamo di utilizzare i cookie con un prefisso `__Host-` se hai bisogno di impostare cookie sensibili nel nome di dominio predefinito per le API API Gateway. Questa pratica ti aiuterà a difendere il tuo dominio dai tentativi CSRF (cross-site request forgery). Per ulteriori informazioni, consulta la pagina Impostazione cookie nella pagina Mozilla Developer Network.

Argomenti

- Crea fasi per le WebSocket API in API Gateway
- Implementa le WebSocket API in API Gateway
- Politica di sicurezza per WebSocket APIs in API Gateway
- Nomi di dominio personalizzati per le WebSocket API in API Gateway

Crea fasi per le WebSocket API in API Gateway

Una fase API è un riferimento logico a uno stato del ciclo di vita dell'API (ad esempio, `dev`, `prod`, `beta` o `v2`). Le fasi API sono identificate dal rispettivo ID API e dal nome della fase e sono incluse nell'URL utilizzato per richiamare l'API. Ogni fase è un riferimento con nome a una distribuzione dell'API e viene resa disponibile per le applicazioni client da chiamare.

Una distribuzione è uno snapshot della configurazione dell'API. Dopo essere stata distribuita a una fase, l'API è disponibile per i client da richiamare. È necessario distribuire un'API per attivare le modifiche apportate.

Variabili di fase

Le variabili di fase sono coppie chiave-valore che è possibile definire per una fase di un'API. WebSocket Fungono da variabili di ambiente e possono essere utilizzate nella configurazione dell'API.

Ad esempio, puoi definire una variabile di fase e quindi impostare il suo valore come un endpoint HTTP per un'integrazione proxy HTTP. Successivamente, puoi fare riferimento all'endpoint utilizzando il nome della variabile di fase associata. In questo modo, puoi utilizzare la stessa configurazione API con un endpoint diverso in ogni fase. Allo stesso modo, puoi utilizzare le variabili di fase per specificare un'integrazione di AWS Lambda funzioni diversa per ogni fase dell'API.

> **Note**
>
> Le variabili di fase non sono destinate ad essere utilizzate per i dati sensibili, come le credenziali. Per trasferire dati sensibili alle integrazioni, usa un AWS Lambda autorizzatore. È possibile passare dati sensibili alle integrazioni nell'output del provider di autorizzazioni Lambda. Per ulteriori informazioni, consulta the section called “Formato della risposta dell'autorizzazione”.

Esempi

Per utilizzare una variabile di fase per personalizzare l'endpoint di integrazione HTTP, è necessario innanzitutto impostare il nome e il valore della variabile di fase (ad esempio `url`) con un valore pari a `example.com`. Quindi, impostare un'integrazione proxy HTTP. Anziché inserire l'URL dell'endpoint, è possibile comunicare ad API Gateway di usare il valore della variabile di fase, ossi, **`http://${stageVariables.url}`**. Questo valore indica ad API Gateway di sostituire la variabile di fase `${}` al runtime, a seconda della fase dell'API.

È possibile fare riferimento alle variabili di fase in modo simile per specificare il nome di una funzione Lambda o un ruolo AWS ARN.

Quando si specifica un nome di funzione Lambda come valore della variabile di fase, è necessario configurare manualmente le autorizzazioni per la funzione Lambda. Per eseguire questa operazione, puoi utilizzare AWS Command Line Interface (AWS CLI).

```
aws lambda add-permission --function-name arn:aws:lambda:XXXXXX:your-lambda-function-
name --source-arn arn:aws:execute-api:us-east-1:YOUR_ACCOUNT_ID:api_id/*/HTTP_METHOD/
resource --principal apigateway.amazonaws.com --statement-id apigateway-access --action
 lambda:InvokeFunction
```

Riferimento alle variabili di fase di API Gateway

URI di integrazione HTTP

Puoi utilizzare una variabile di fase come parte di un URI di integrazione HTTP, come mostrato negli esempi seguenti.

- Un URI completo senza protocoll – `http://${stageVariables.<variable_name>}`
- Un dominio completo – `http://${stageVariables.<variable_name>}/resource/operation`
- Un sottodominio – `http://${stageVariables.<variable_name>}.example.com/resource/operation`
- Un percorso – `http://example.com/${stageVariables.<variable_name>}/bar`
- Una stringa di query – `http://example.com/foo?q=${stageVariables.<variable_name>}`

Funzioni Lambda

Puoi utilizzare una variabile di fase al posto di un nome di funzione o alias Lambda, come illustrato negli esempi seguenti.

- `arn:aws:apigateway:<region>:lambda:path/2015-03-31/functions/arn:aws:lambda:<region>:<account_id>:function:${stageVariables.<function_variable_name>}/invocations`
- `arn:aws:apigateway:<region>:lambda:path/2015-03-31/functions/arn:aws:lambda:<region>:<account_id>:function:<function_name>:${stageVariables.<version_variable_name>}/invocations`

> **Note**
>
> Per utilizzare una variabile di fase per una funzione Lambda, la funzione deve essere nello stesso account dell'API. Le variabili di fase non supportano le funzioni Lambda tra più account.

AWS credenziali di integrazione

È possibile utilizzare una variabile stage come parte di un ARN di credenziali AWS utente o di ruolo, come illustrato nell'esempio seguente.

- `arn:aws:iam::<account_id>:${stageVariables.<variable_name>}`

Implementa le WebSocket API in API Gateway

Dopo aver creato l' WebSocket API, devi distribuirla per renderla disponibile per essere richiamata dagli utenti.

Per distribuire un'API, crea una distribuzione API e associala a una fase. Ogni fase è una snapshot dell'API ed è resa disponibile per essere chiamata dalle app client.

> **Important**
>
> Ogni volta che aggiorni un'API, devi ripeterne l'implementazione. Le modifiche a qualcosa di diverso dalle impostazioni di fase richiedono una nuova implementazione, incluse le modifiche alle seguenti risorse:
>
> - Route
> - Integrazioni
> - Authorizers
>
> C'è un limite predefinito di 10 fasi per API. Si consiglia di riutilizzare le fasi per le implementazioni.

Per chiamare un' WebSocket API distribuita, il client invia un messaggio all'URL dell'API. L'URL viene determinato dal nome host e dal nome fase dell'API.

> **Note**
>
> API Gateway supporta payload fino a 128 KB con dimensione del frame massima di 32 KB. Se un messaggio supera i 32 KB, deve essere suddiviso in più frame, ciascuno di 32 KB o più piccolo.

Utilizzando il nome di dominio predefinito dell'API, l'URL di (ad esempio) un' WebSocket API in una determinata fase (*{stageName}*) ha il seguente formato:

```
wss://{api-id}.execute-api.{region}.amazonaws.com/{stageName}
```

Per rendere l'URL dell' WebSocket API più intuitivo, puoi creare un nome di dominio personalizzato (ad esempio,`api.example.com`) per sostituire il nome host predefinito dell'API. Il processo di configurazione è lo stesso delle API REST. Per ulteriori informazioni, consulta the section called "Nomi di dominio personalizzati".

Le fasi permettono un efficace controllo delle versioni dell'API. Ad esempio, puoi distribuire un'API in una fase `test` e una fase `prod` e utilizzare la fase `test` come build di test e la fase `prod` come build stabile. Dopo che gli aggiornamenti hanno superato il test, puoi promuovere la fase `test` alla fase `prod`. La promozione può essere eseguita ridistribuendo l'API nella fase `prod`. Per ulteriori dettagli sulle fasi, consulta the section called "Configurare una fase".

Argomenti

- Crea una distribuzione WebSocket dell'API utilizzando il AWS CLI
- Creare una distribuzione WebSocket API utilizzando la console API Gateway

Crea una distribuzione WebSocket dell'API utilizzando il AWS CLI

AWS CLI Per creare una distribuzione, utilizzate il comando create-deployment come illustrato nell'esempio seguente:

```
aws apigatewayv2 --region us-east-1 create-deployment --api-id aabbccddee
```

Output di esempio:

```
{
    "DeploymentId": "fedcba",
```

```
    "DeploymentStatus": "DEPLOYED",
    "CreatedDate": "2018-11-15T06:49:09Z"
}
```

L'API distribuita non può essere chiamata fino a quando non si associa la distribuzione a una fase. Puoi creare una nuova fase o riutilizzare una fase creata in precedenza.

Per creare una nuova fase e associarla alla distribuzione, utilizzate il comando create-stage come mostrato nell'esempio seguente:

```
aws apigatewayv2 --region us-east-1 create-stage --api-id aabbccddee --deployment-id
 fedcba --stage-name test
```

Output di esempio:

```
{
    "StageName": "test",
    "CreatedDate": "2018-11-15T06:50:28Z",
    "DeploymentId": "fedcba",
    "DefaultRouteSettings": {
        "MetricsEnabled": false,
        "ThrottlingBurstLimit": 5000,
        "DataTraceEnabled": false,
        "ThrottlingRateLimit": 10000.0
    },
    "LastUpdatedDate": "2018-11-15T06:50:28Z",
    "StageVariables": {},
    "RouteSettings": {}
}
```

Per riutilizzare una fase esistente, aggiorna la `deploymentId` proprietà della fase con l'ID di distribuzione appena creato (*`{deployment-id}`*) utilizzando il comando update-stage.

```
 aws apigatewayv2 update-stage --region {region} \
    --api-id {api-id} \
    --stage-name {stage-name} \
    --deployment-id {deployment-id}
```

Creare una distribuzione WebSocket API utilizzando la console API Gateway

Per utilizzare la console API Gateway per creare una distribuzione per un' WebSocket API:

1. Accedere alla console API Gateway e scegliere l'API.
2. Seleziona Deploy API (Distribuisci API).
3. Scegliere la fase desiderata dall'elenco a discesa oppure immettere il nome di una nuova fase.

Politica di sicurezza per WebSocket APIs in API Gateway

APIGateway applica una politica di sicurezza `TLS_1_2` per tutti gli WebSocket API endpoint.

Una policy di sicurezza è una combinazione predefinita di TLS versioni minime e suite di crittografia offerte da Amazon Gateway. API Il TLS protocollo risolve problemi di sicurezza della rete come la manomissione e l'intercettazione tra client e server. Quando i client stabiliscono una TLS stretta di mano nei vostri confronti API tramite il dominio personalizzato, la politica di sicurezza applica la TLS versione e le opzioni della suite di crittografia che i clienti possono scegliere di utilizzare. Questa politica di sicurezza accetta il traffico TLS 1.2 e TLS 1.3 e rifiuta il traffico 1.0. TLS

TLSProtocolli e cifrari supportati per WebSocket APIs

La tabella seguente descrive i TLS protocolli supportati per. WebSocket APIs

TLSprotocolli	TLS_1_2 politica di sicurezza
TLSv1.3	Sì
TLSv12.	Sì

La tabella seguente descrive i codici TLS disponibili per la politica di sicurezza TLS 1_2 per. WebSocket APIs

TLScifrari	TLS_1_2 politica di sicurezza
TLS_ _128_ _ AES GCM SHA256	Sì

TLScifrari	TLS_1_2 politica di sicurezza
TLS_ _256_ _ AES GCM SHA384	Sì
TLS_ 0_ 05_ CHACHA2 POLY13 SHA256	Sì
ECDHE-ECDSA-AES128-GCM-SHA256	Sì
ECDHE-RSA-AES128-GCM-SHA256	Sì
ECDHE-ECDSA-AES128-SHA256	Sì
ECDHE-RSA-AES128-SHA256	Sì
ECDHE-ECDSA-AES256-GCM-SHA384	Sì
ECDHE-RSA-AES256-GCM-SHA384	Sì
ECDHE-ECDSA-AES256-SHA384	Sì
ECDHE-RSA-AES256-SHA384	Sì
AES128-GCM-SHA256	Sì
AES128-SHA256	Sì

TLScifrari	TLS_1_2 politica di sicurezza
AES256-GCM-SHA384	Sì
AES256-SHA256	Sì

Nomi aperti SSL e cifrati RFC

Open SSL e IETF RFC 5246, usano nomi diversi per gli stessi cifrari. Per un elenco dei nomi dei cifrari, vedere. the section called “Nomi aperti SSL e cifrati RFC”

Informazioni su e REST APIs HTTP APIs

Per ulteriori informazioni su REST APIs e HTTPAPIs, vedere the section called “Scegli una politica di sicurezza” ethe section called “Politica di sicurezza per HTTP APIs in API Gateway”.

Nomi di dominio personalizzati per le WebSocket API in API Gateway

I nomi di dominio personalizzati sono URL più semplici e più intuitivi che è possibile fornire agli utenti delle API.

Dopo aver distribuito un'API, tu e i tuoi clienti potete richiamarla usando l'URL di base predefinito nel formato seguente:

```
https://api-id.execute-api.region.amazonaws.com/stage
```

dove l'*`api-id`* è generato da API Gateway, la *`regione`* è la AWS regione e la *`fase`* viene specificata dall'utente al momento della distribuzione dell'API.

La parte dell'URL relativa al nome host *`api-id`*`.execute-api.`*`region`*`.amazonaws.com` si riferisce a un endpoint API. Il nome dell'endpoint dell'API predefinito viene generato in modo casuale, è difficile da richiamare e non è intuitivo.

Con i nomi di dominio personalizzati, è possibile impostare il nome host dell'API e scegliere un percorso base (ad esempio `myservice`) per mappare l'URL alternativo all'API. Ad esempio, un URL di base dell'API più intuitivo può diventare:

```
https://api.example.com/myservice
```

Considerazioni

Le seguenti considerazioni potrebbero influire sull'utilizzo di un nome di dominio personalizzato.

- Se mappi un nome di dominio personalizzato a un' WebSocket API, non puoi mapparlo a un'API REST o a un'API HTTP.
- Sono supportati solo i nomi di dominio personalizzati regionali.
- Per la versione TLS minima, è supportato solo TLS 1.2.
- È necessario creare o aggiornare il record di risorse del provider DNS da mappare all'endpoint API. In assenza di questa mappatura, le richieste API destinate al nome di dominio personalizzato non possono raggiungere API Gateway.
- Puoi supportare un numero quasi infinito di nomi di dominio senza superare la quota predefinita utilizzando un certificato wildcard. Per ulteriori informazioni, consulta the section called “Nomi di dominio personalizzati con caratteri jolly”.

Prerequisiti

Di seguito sono riportati i prerequisiti per un nome di dominio personalizzato.

Registrare un nome di dominio

Per configurare nomi di dominio personalizzati per le API, devi avere un nome di dominio Internet registrato. Puoi registrare il tuo nome di dominio Internet utilizzando Amazon Route 53 o utilizzando un registrar di domini di terze parti a tua scelta. Il tuo nome di dominio personalizzato può essere il nome di un sottodominio o il dominio principale (noto anche come «apice di zona») di un dominio Internet registrato.

Il nome di dominio deve seguire le specifiche RFC 1035 e può avere un massimo di 63 ottetti per etichetta e 255 ottetti in totale.

Certificati per nomi di dominio personalizzati

Prima di configurare un nome di dominio personalizzato per un'API, devi disporre di un certificato SSL/TLS pronto in ACM. Se ACM non è disponibile nella AWS regione in cui stai creando il tuo nome di dominio personalizzato, devi importare un certificato in API Gateway in quella regione.

Per importare un certificato SSL/TLS, devi fornire il corpo del certificato SSL/TLS in formato PEM, la sua chiave privata e la catena di certificati per il nome di dominio personalizzato.

Ogni certificato archiviato in ACM è identificato dal relativo ARN. Con i certificati emessi da ACM, non devi preoccuparti di esporre dettagli sensibili del certificato, come la chiave privata. Per utilizzare un certificato AWS gestito per un nome di dominio, è sufficiente fare riferimento al relativo ARN.

Se l'applicazione utilizza il blocco dei certificati, a volte noto come pinning SSL, per bloccare un certificato ACM, l'applicazione potrebbe non essere in grado di connettersi al dominio dopo il rinnovo del certificato. AWS Per ulteriori informazioni, consulta Probemi nel blocco dei certificati nella Guida per l'utente di AWS Certificate Manager .

Nomi di dominio personalizzati con caratteri jolly

Con i nomi di dominio personalizzati con caratteri jolly, è possibile supportare un numero quasi infinito di nomi di dominio senza superare la quota predefinita. Ad esempio, è possibile dare a ciascuno dei clienti il proprio nome di dominio, `customername.api.example.com`.

Per creare un nome di dominio personalizzato con caratteri jolly, specificare un carattere jolly (`*`) come primo sottodominio di un dominio personalizzato che rappresenta tutti i possibili sottodomini di un dominio radice.

Ad esempio, il nome di dominio personalizzato con caratteri jolly `*.example.com` genera sottodomini quali `a.example.com`, `b.example.com` e `c.example.com`, indirizzati tutti allo stesso dominio.

I nomi di dominio personalizzati con caratteri jolly supportano configurazioni distinte dai nomi di dominio personalizzati standard di API Gateway. Ad esempio, in un singolo AWS account, è possibile configurare e comportarsi in modo diverso. `*.example.com` `a.example.com`

È possibile utilizzare le variabili di contesto `$context.domainName` e `$context.domainPrefix` per determinare il nome di dominio utilizzato da un client per chiamare l'API. Per ulteriori informazioni sulle variabili di contesto, consulta APIModello di mappatura del gateway e riferimento alla variabile di registrazione degli accessi.

Per creare un nome di dominio personalizzato con caratteri jolly, è necessario fornire un certificato emesso da ACM che sia stato convalidato utilizzando il DNS o il metodo di convalida della posta elettronica.

> **Note**
>
> Non puoi creare un nome di dominio personalizzato con caratteri jolly se un altro AWS account ha creato un nome di dominio personalizzato che è in conflitto con il nome di dominio personalizzato con caratteri jolly. Ad esempio, se l'account A ha creato `a.example.com`, l'account B non può creare il nome di dominio personalizzato con caratteri jolly `*.example.com`.
> Se l'account A e l'account B condividono un proprietario, è possibile contattare il Centro assistenza AWS per richiedere un'eccezione.

Passaggi successivi per i nomi di dominio personalizzati

Per configurare un nome di dominio personalizzato per un'API HTTP, utilizza la documentazione della sezione API REST della Guida per sviluppatori di API Gateway.

Innanzitutto, specifica un certificato per il tuo nome di dominio personalizzato. Per ulteriori informazioni, consulta the section called “Prepara i certificati in AWS Certificate Manager”. Successivamente, crei un nome di dominio personalizzato regionale. Per ulteriori informazioni, consulta the section called “Configura un nome di dominio personalizzato regionale”.

Mappa le fasi dell'API su un nome di dominio personalizzato per le WebSocket API

È possibile utilizzare le mappature API per connettere le fasi API a un nome di dominio personalizzato. Dopo aver creato un nome di dominio e aver configurato i record DNS, è possibile utilizzare le mappature API per inviare il traffico alle API tramite il nome di dominio personalizzato.

Una mappatura API specifica un'API, una fase e, facoltativamente, un percorso da utilizzare per la mappatura. Ad esempio, è possibile mappare la fase `production` di un'API su `wss://api.example.com/orders`.

Prima di creare una mappatura API, è necessario disporre di un'API, di una fase e di un nome di dominio personalizzato. Per ulteriori informazioni sulla creazione di un nome di dominio personalizzato, consulta the section called “Configura un nome di dominio personalizzato regionale”.

Restrizioni

- In una mappatura API, il nome di dominio personalizzato e le API mappate devono trovarsi nello stesso account. AWS
- Le mappature API devono contenere solo lettere, numeri e i seguenti caratteri: `$-_.+!*'()`.

- La lunghezza massima per il percorso in una mappatura API è di 300 caratteri.
- Non WebSocket è possibile mappare le API allo stesso nome di dominio personalizzato di un'API HTTP o di un'API REST.

Creare una mappatura API

Per creare una mappatura API, innanzitutto è necessario creare un nome di dominio personalizzato, un'API e una fase. Per informazioni sulla creazione di un nome di dominio personalizzato, consulta the section called “Configura un nome di dominio personalizzato regionale”.

AWS Management Console

Per creare una mappatura API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere Nomi di dominio personalizzati.
3. Selezionare un nome di dominio personalizzato già creato.
4. Scegliere API mappings (mappature API).
5. Scegliere Configure API mappings (Configura mappature API).
6. Scegliere Add new mapping (Aggiungi nuova mappatura).
7. Immettere un'API, uno Stage (Fase)e, facoltativamente, un Path (Percorso).
8. Selezionare Salva.

AWS CLI

Il AWS CLI comando seguente crea una mappatura delle API. In questo esempio, API Gateway invia le richieste `api.example.com/v1` all'API e alla fase specificate.

```
aws apigatewayv2 create-api-mapping \
    --domain-name api.example.com \
    --api-mapping-key v1 \
    --api-id a1b2c3d4 \
    --stage test
```

AWS CloudFormation

L' AWS CloudFormation esempio seguente crea una mappatura delle API.

```
MyApiMapping:
  Type: 'AWS::ApiGatewayV2::ApiMapping'
  Properties:
    DomainName: api.example.com
    ApiMappingKey: 'v1'
    ApiId: !Ref MyApi
    Stage: !Ref MyStage
```

Disabilita l'endpoint predefinito per le API WebSocket

Per impostazione predefinita, i client possono richiamare l'API utilizzando l'endpoint `execute-api` generato da API Gateway per l'API. Per garantire che i client possano accedere all'API solo utilizzando un nome di dominio personalizzato con l'autenticazione TLS reciproca, disattivare l'endpoint `execute-api` predefinito. Quando si disattiva l'endpoint predefinito, questa operazione influisce su tutte le fasi di un'API.

La procedura seguente mostra come disabilitare l'endpoint predefinito per un' WebSocket API.

AWS Management Console

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli un' WebSocket API.
3. Scegli Impostazioni API.
4. Nei dettagli dell'API, scegli Modifica.
5. Per Endpoint predefinito, seleziona Inattivo.
6. Seleziona Salvataggio delle modifiche.
7. Nel riquadro di navigazione principale, scegli Percorsi.
8. Scegli Deploy, quindi ridistribuisci l'API o crea una nuova fase per rendere effettiva la modifica.

AWS CLI

Il AWS CLI comando seguente disabilita l'endpoint predefinito per un'API. WebSocket

```
aws apigatewayv2 update-api \
    --api-id abcdef123 \
```

```
    --disable-execute-api-endpoint
```

Dopo aver disabilitato l'endpoint predefinito, è necessario distribuire l'API per rendere effettiva la modifica.

Il AWS CLI comando seguente crea una distribuzione.

```
aws apigatewayv2 create-deployment \
    --api-id abcdef123 \
    --stage-name dev
```

Proteggi le tue WebSocket API in API Gateway

È possibile configurare la limitazione della larghezza di banda della rete per le API per proteggerle da un numero eccessivo di richieste. Le limitazioni della larghezza di banda della rete sono applicate sulla base del massimo sforzo e dovrebbero essere considerate come obiettivi e non limiti massimi garantiti delle richieste.

API Gateway limita la larghezza di banda della rete delle richieste nell'API mediante l'algoritmo di token bucket, dove un token rappresenta una richiesta. Nello specifico, API Gateway esamina il tasso e i picchi di invio di richieste per tutte le API dell'account, per regione. Nell'algoritmo di bucket token, un picco può consentire il superamento predefinito di tali limiti, ma anche altri fattori possono causare il superamento dei limiti in alcuni casi.

Quando le richieste inviate superano i limiti impostati per il tasso a regime e per i limiti di picco, API Gateway inizia a limitare le richieste. I clienti potrebbero ricevere risposte agli errori `429 Too Many Requests` a questo punto. Quando rileva queste eccezioni, il client può reinviare le richieste non riuscite in modo da limitare la velocità.

In qualità di sviluppatore dell'API, puoi impostare i limiti per le singole fasi o le singole route dell'API per migliorare le prestazioni generali in tutte le API dell'account.

Limitazione a livello di account per regione

Per impostazione predefinita, API Gateway limita il numero di richieste con stato costante al secondo (RPS) per tutte le API all'interno di un account AWS , per Regione. Limita anche i picchi, ovvero la dimensione massima del bucket, per tutte le API di un account AWS , per regione. In API Gateway, il limite dei picchi rappresenta il numero massimo di invii di richieste che API Gateway può gestire

prima di restituire risposte di errore `429 Too Many Requests`. Per ulteriori informazioni sulla limitazione delle quote, vedere Quote e note importanti.

I limiti per account vengono applicati a tutte le API di un account in una regione specificata. Il limite del tasso a livello di account può essere aumentato - sono possibili limiti più elevati con API con timeout più brevi e payload più piccoli. Per richiedere un aumento dei limiti di limitazione a livello di account per Regione, contatta il Centro assistenza AWS. Per ulteriori informazioni, consulta Quote e note importanti. Tieni presente che questi limiti non possono essere superiori ai limiti di AWS limitazione.

Throttling a livello di instradamento

Puoi impostare il throttling a livello di route per sostituire i limiti di throttling delle richieste a livello di account per una fase specifica o per singoli route nell'API. I limiti della limitazione (della larghezza di banda della rete) della route di default non possono superare i limiti di velocità a livello di account.

È possibile configurare la limitazione a livello di percorso utilizzando AWS CLI. Il comando seguente configura la limitazione personalizzata per lo stadio e la route specificati di un'API.

```
aws apigatewayv2 update-stage \
    --api-id a1b2c3d4 \
    --stage-name dev \
    --route-settings '{"messages":
{"ThrottlingBurstLimit":100,"ThrottlingRateLimit":2000}}'
```

Monitora WebSocket le API in API Gateway

Puoi utilizzare CloudWatch metriche e CloudWatch registri per monitorare le API. WebSocket Combinando log e parametri, puoi registrare gli errori e monitorare le prestazioni dell'API.

> **Note**
>
> L'API Gateway potrebbe non generare log e parametri nei seguenti casi:
>
> - Errori 413 per richiesta troppo grande
> - Errori 429 per troppe richieste
> - Errori serie 400 per richieste inviate a un dominio personalizzato che non dispone del mapping API

- Errori serie 500 per malfunzionamenti interni

Argomenti

- Monitora WebSocket API l'esecuzione con CloudWatch metriche
- Configurare la registrazione per WebSocket APIs Gateway API

Monitora WebSocket API l'esecuzione con CloudWatch metriche

Puoi utilizzare i CloudWatch parametri di Amazon per monitorare WebSocket APIs. La configurazione è simile a quella utilizzata per RESTAPIs. Per ulteriori informazioni, consulta Monitora REST API l'esecuzione con i CloudWatch parametri di Amazon.

Le seguenti metriche sono supportate per WebSocket APIs:

Parametro	Descrizione
ConnectCount	Il numero di messaggi inviati all'integrazione route `$connect`.
MessageCount	Il numero di messaggi inviati al WebSocket API, da o verso il client.
IntegrationError	Il numero di richieste che restituiscono una risposta 4XX/5XX dall'integrazione.
ClientError	Il numero di richieste con una risposta 4XX restituita da API Gateway prima che l'integra zione venga richiamata.
ExecutionError	Errori che si sono verificati durante la chiamata all'integ razione.

Parametro	Descrizione
IntegrationLatency	La differenza di tempo tra API Gateway che invia la richiesta all'integrazione e API Gateway che riceve la risposta dall'inte grazione. Soppresso per callback e integrazioni fittizie.

È possibile utilizzare le dimensioni nella tabella seguente per filtrare le metriche di API Gateway.

Dimensione	Descrizione
ApiId	Filtra le metriche di API Gateway per un utente API con l'ID specificatoAPI.
ApiId, Fase	Filtra le metriche di API Gateway per uno API stage con l'APIID e l'ID dello stage specificati.
ApiId, Metodo, risorsa, fase	Filtra le metriche del API gateway per un API metodo con l'ID, l'APIID dello stadio, il percorso di risorsa e l'ID del percorso specificati. APIGateway non invierà queste metriche a meno che tu non abbia abilitato esplicita mente le metriche dettaglia te. CloudWatch È possibile farlo richiamando l'UpdateSta geazione del API Gateway V2 REST API a cui aggiornar e la proprietà. `detailedM`

Dimensione	Descrizione
	`etricsEnabled` `true` In alternativa, è possibile chiamare il AWS CLI comando update-stage per aggiornare la `DetailedMetricsEnabled` proprietà a. `true` L'abilitazione di tali parametri comporta addebiti aggiuntivi sul tuo account. Per informazioni sui prezzi, consulta la pagina CloudWatchdei prezzi di Amazon.

Configurare la registrazione per WebSocket APIs Gateway API

È possibile abilitare la registrazione per scrivere i log nei registri. CloudWatch Esistono due tipi di API registrazione CloudWatch: registrazione dell'esecuzione e registrazione degli accessi. Nella registrazione dell'esecuzione, API Gateway gestisce i registri. CloudWatch Il processo include la creazione di gruppi e flussi di log e la segnalazione ai flussi di log delle richieste e delle risposte dell'intermediario.

Nella registrazione degli accessi, tu, in qualità di API sviluppatore, desideri registrare chi ha effettuato l'accesso al tuo API e in che modo il chiamante ha effettuato l'accesso al. API Puoi creare il tuo gruppo di log o scegliere un gruppo di log esistente che possa essere gestito da API Gateway. Per specificare i dettagli di accesso, seleziona variabili `$context` (espresse in un formato a scelta) e scegli un gruppo di log come destinazione.

Per istruzioni su come configurare la CloudWatch registrazione, vederethe section called “Configurare la CloudWatch API registrazione utilizzando la console API Gateway”.

Quando specifichi il Log Format (Formato di log), puoi scegliere quali variabili di contesto registrare. Sono supportate le seguenti variabili.

Parametro	Descrizione
`$context.apiId`	L'identificatore che API Gateway assegna al tuo. API
`$context.authorize.error`	Il messaggio di errore di autorizzazione.
`$context.authorize.latency`	La latenza di autorizzazione in ms.
`$context.authorize.status`	Il codice di stato restituito da un tentativo di autorizzazione.
`$context.authorizer.error`	Il messaggio di errore restituito da un'autori zzazione.
`$context.authorizer.integra tionLatency`	La latenza del provider di autorizzazioni Lambda in ms.
`$context.authorizer.integra tionStatus`	Il codice di stato restituito da un'autorizzazione Lambda.
`$context.authorizer.latency`	La latenza di autorizzazione in ms.
`$context.authorizer.requestId`	L'ID della AWS richiesta dell'endpoint.
`$context.authorizer.status`	Il codice di stato restituito da un'autorizzazione.
`$context.authorizer.principalId`	L'identificazione dell'utente principale associata al token inviato dal client e restituito da una funzione Lambda dell'autorizzazione API Gateway Lambda. (Un provider di autorizza zioni Lambda in precedenza noto come autorizzazioni ad hoc).
`$context.authorizer.` *`property`*	Il valore stringato della coppia chiave-valore specificata della `context` mappa restituito da una funzione di autorizzazione Gateway API Lambda. Ad esempio, se le autorizzazioni restituiscono la mappa `context` seguente:

Parametro	Descrizione
	`"context" : { "key": "value", "numKey": 1, "boolKey": true }` la chiamata di `$context.authorizer.key` restituisce la stringa `"value"`, la chiamata di `$context.authorizer.numKey` restituisce la stringa `"1"` e la chiamata di `$context.authorizer.boolKey` restituisce la stringa `"true"`.
`$context.authenticate.error`	Il messaggio di errore restituito da un tentativo di autenticazione.
`$context.authenticate.latency`	La latenza di autenticazione in ms.
`$context.authenticate.status`	Il codice di stato restituito da un tentativo di autenticazione.
`$context.connectedAt`	L'ora della connessione in formato Epoch.
`$context.connectionId`	Un ID univoco per la connessione, che può essere utilizzato per effettuare un callback al client.
`$context.domainName`	Un WebSocket API nome di dominio per. Può essere utilizzato per effettuare un callback al client (invece di un valore codificato).
`$context.error.message`	Una stringa che contiene un messaggio di errore API Gateway.

Parametro	Descrizione
`$context.error.messageString`	Valore `$context.error.message` tra virgolette, ovvero `"$context.error.message"` .
`$context.error.responseType`	Il tipo di risposta di errore.
`$context.error.validationErrorString`	Stringa contenente un messaggio di errore di convalida dettagliato.
`$context.eventType`	Il tipo di evento: `CONNECT`, `MESSAGE` o `DISCONNECT` .
`$context.extendedRequestId`	Equivalente a `$context.requestId` .
`$context.identity.accountId`	L'ID AWS dell'account associato alla richiesta.
`$context.identity.apiKey`	La chiave API del proprietario associata alla richiesta con chiave abilitataAPI.
`$context.identity.apiKeyId`	L'ID della API chiave associato alla richiesta abilitata alla chiave API
`$context.identity.caller`	Identificatore dell'entità principale dell'intermediario che ha firmato la richiesta. Supportato per i percorsi che utilizzano l'IAMautorizzazione.

Parametro	Descrizione
`$context.identity.cognitoAuthenticationProvider`	Un elenco separato da virgole di tutti i provider di autenticazione Amazon Cognito utilizzati dal chiamante che effettua la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito. Ad esempio, per un'identità di un pool di utenti Amazon Cognito, `cognito-idp.`*`region`*`.amazonaws.com/`*`user_pool_id`*`,cognito-idp.`*`region`*`.amazonaws.com/`*`user_pool_id`*`:CognitoSignIn:`*`token subject claim`* Per informazioni sui provider di autenticazione di Amazon Cognito disponibili, consulta Using Federated Identities nella Amazon Cognito Developer Guide.
`$context.identity.cognitoAuthenticationType`	Tipo di autenticazione Amazon Cognito dell'intermediario da cui proviene la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito. I valori possibili includono `authenticated` per le identità autenticate e `unauthenticated` per le identità non autenticate.
`$context.identity.cognitoIdentityId`	ID identità di Amazon Cognito dell'intermediario da cui proviene la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito.
`$context.identity.cognitoIdentityPoolId`	ID pool di identità di Amazon Cognito dell'intermediario da cui proviene la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito.

Parametro	Descrizione
`$context.identity.principalOrgId`	L'ID organizzazione AWS. Supportato per i percorsi che utilizzano l'autorizzazione. IAM
`$context.identity.sourceIp`	L'indirizzo IP di origine della TCP connessione che effettua la richiesta a API Gateway.
`$context.identity.user`	Identificatore dell'entità principale dell'utente che sarà autorizzato per l'accesso alle risorse. Supportato per i percorsi che utilizzano IAM l'autorizzazione.
`$context.identity.userAgent`	L'agente utente del API chiamante.
`$context.identity.userArn`	L'Amazon Resource Name (ARN) dell'utente effettivo identificato dopo l'autenticazione.
`$context.integration.error`	Il messaggio di errore restituito da un'integr azione.
`$context.integration.integr ationStatus`	Per l'integrazione del proxy Lambda, il codice di stato restituito dal codice della funzione Lambda di backend AWS Lambda, non dal codice della funzione Lambda.
`$context.integration.latency`	Latenza di integrazione in ms. Equivalente a `$context.integrationLatency` .
`$context.integration.requestId`	L'ID della AWS richiesta dell'endpoint. Equivalente a `$context.awsEndpoi ntRequestId` .
`$context.integration.status`	Il codice di stato restituito da un'integrazione. Per le integrazioni proxy Lambda, questo è il codice di stato restituito dal codice della funzione Lambda. Equivalente a `$context. integrationStatus` .

Parametro	Descrizione
`$context.integrationLatency`	La latenza di integrazione in millisecondi, disponibile solo per il log di accesso.
`$context.messageId`	Un ID univoco sul lato server per un messaggio . Disponibile solo quando `$context.eventType` è `MESSAGE`.
`$context.requestId`	Come `$context.extendedRequestId` .
`$context.requestTime`	L'ora della richiesta in CLFformato -formattato (). `dd/MMM/yyyy:HH:mm:ss +-hhmm`
`$context.requestTimeEpoch`	L'ora della richiesta in formato epoca (Unix epoch) in millisecondi.
`$context.routeKey`	La chiave di instradamento selezionata.
`$context.stage`	La fase di distribuzione della API chiamata (ad esempio, beta o prod).
`$context.status`	Lo stato della risposta.
`$context.waf.error`	Il messaggio di errore restituito da AWS WAF.
`$context.waf.latency`	La AWS WAF latenza in ms.
`$context.waf.status`	Il codice di stato restituito da AWS WAF.

Esempi di alcuni formati di log di accesso di uso comune sono mostrati nella console API Gateway e sono elencati di seguito.

- `CLF` (Common Log Format):

```
$context.identity.sourceIp $context.identity.caller \
$context.identity.user [$context.requestTime] "$context.eventType $context.routeKey
 $context.connectionId" \
$context.status $context.requestId
```

I caratteri di continuazione (\) sono intesi come un aiuto visivo. Il formato del registro deve essere una singola riga. È possibile aggiungere un carattere di nuova riga (\n) alla fine del formato di registro per includere una nuova riga alla fine di ogni voce di registro.

- `JSON`:

```
{
"requestId":"$context.requestId", \
"ip": "$context.identity.sourceIp", \
"caller":"$context.identity.caller", \
"user":"$context.identity.user", \
"requestTime":"$context.requestTime", \
"eventType":"$context.eventType", \
"routeKey":"$context.routeKey", \
"status":"$context.status", \
"connectionId":"$context.connectionId"
}
```

I caratteri di continuazione (\) sono intesi come un aiuto visivo. Il formato del registro deve essere una singola riga. È possibile aggiungere un carattere di nuova riga (\n) alla fine del formato di registro per includere una nuova riga alla fine di ogni voce di registro.

- `XML`:

```
<request id="$context.requestId"> \
 <ip>$context.identity.sourceIp</ip> \
 <caller>$context.identity.caller</caller> \
 <user>$context.identity.user</user> \
 <requestTime>$context.requestTime</requestTime> \
 <eventType>$context.eventType</eventType> \
 <routeKey>$context.routeKey</routeKey> \
 <status>$context.status</status> \
 <connectionId>$context.connectionId</connectionId> \
</request>
```

I caratteri di continuazione (\) sono intesi come un aiuto visivo. Il formato del registro deve essere una singola riga. È possibile aggiungere un carattere di nuova riga (\n) alla fine del formato di registro per includere una nuova riga alla fine di ogni voce di registro.

- `CSV` (valori separati da virgola):

```
$context.identity.sourceIp,$context.identity.caller, \
```

```
$context.identity.user,$context.requestTime,$context.eventType, \
$context.routeKey,$context.connectionId,$context.status, \
$context.requestId
```

I caratteri di continuazione (\) sono intesi come un aiuto visivo. Il formato del registro deve essere una singola riga. È possibile aggiungere un carattere di nuova riga (\n) alla fine del formato di registro per includere una nuova riga alla fine di ogni voce di registro.

Documentazione di riferimento Amazon Resource Name (ARN) API Gateway

Nelle tabelle seguenti sono elencati gli Amazon Resource Name (ARN) delle risorse API Gateway. Per ulteriori informazioni sull'utilizzo degli ARN nelle AWS Identity and Access Management politiche, consulta Come funziona Amazon API Gateway con IAM e. Controlla l'accesso a un REST API con IAM autorizzazioni

API HTTP e risorse WebSocket API

Risorsa	ARN
AccessLogSettings	`arn:partition :apigateway: region::/apis/api-id/stages/stage-name /accesslogsettings`
Api	`arn:partition :apigateway: region::/apis/api-id`
API	`arn:partition :apigateway: region::/apis`
ApiMapping	`arn:partition :apigateway: region::/domainnames/ domain-name /apimappings/ id`
ApiMappings	`arn:partition :apigateway: region::/domainnames/ domain-name /apimappings`
Authorizer	`arn:partition :apigateway: region::/apis/api-id/authorizers/ id`

Risorsa	ARN
Authorizers	`arn:partition :apigateway: region::/apis/api-id/authorizers`
Cors	`arn:partition :apigateway: region::/apis/api-id/cors`
Implementazione	`arn:partition :apigateway: region::/apis/api-id/deployments/ id`
Distribuzioni	`arn:partition :apigateway: region::/apis/api-id/deployments`
DomainName	`arn:partition :apigateway: region::/domainnames/ domain-name`
DomainNames	`arn:partition :apigateway: region::/domainnames`
ExportedAPI	`arn:partition :apigateway: region::/apis/api-id/exports/ specification`
Integrazione	`arn:partition :apigateway: region::/apis/api-id/integrations/ integration-id`
Integrazioni	`arn:partition :apigateway: region::/apis/api-id/integrations`

Risorsa	ARN
IntegrationResponse	arn:*partition* :apigateway: *region*::/apis/*api-id*/integrationresponses/ *integration-response*
IntegrationResponses	arn:*partition* :apigateway: *region*::/apis/*api-id*/integrationresponses
Modello	arn:*partition* :apigateway: *region*::/apis/*api-id*/models/*id*
Modelli	arn:*partition* :apigateway: *region*::/apis/*api-id*/models
ModelTemplate	arn:*partition* :apigateway: *region*::/apis/*api-id*/models/*id*/template
Route	arn:*partition* :apigateway: *region*::/apis/*api-id*/routes/*id*
Route	arn:*partition* :apigateway: *region*::/apis/*api-id*/routes
RouteRequestParameter	arn:*partition* :apigateway: *region*::/apis/*api-id*/routes/*id*/requestparameters/ *key*
RouteResponse	arn:*partition* :apigateway: *region*::/apis/*api-id*/routes/*id*/routeresponses/ *id*
RouteResponses	arn:*partition* :apigateway: *region*::/apis/*api-id*/routes/*id*/routeresponses

Risorsa	ARN
RouteSettings	`arn:partition :apigateway: region::/apis/api-id/stages/stage-name /routesettings/ route-key`
Stage	`arn:partition :apigateway: region::/apis/api-id/stages/stage-name`
Stage	`arn:partition :apigateway: region::/apis/api-id/stages`
VpcLink	`arn:partition :apigateway: region::/vpclinks/ vpclink-id`
VpcLinks	`arn:partition :apigateway: region::/vpclinks`

Risorse API REST

Risorsa	ARN
Account	`arn:partition :apigateway: region::/account`
ApiKey	`arn:partition :apigateway: region::/apikeys/ id`
ApiKeys	`arn:partition :apigateway: region::/apikeys`
Authorizer	`arn:partition :apigateway: region::/restapis/ api-id/authorizers/ id`

Risorsa	ARN
Authorizers	`arn:partition :apigateway: region::/restapis/ api-id/authorizers`
BasePathMapping	`arn:partition :apigateway: region::/domainnames/ domain-name /basepathmappings/ basepath`
BasePathMappings	`arn:partition :apigateway: region::/domainnames/ domain-name /basepathmappings`
ClientCertificate	`arn:partition :apigateway: region::/clientcertificates/ id`
ClientCertificates	`arn:partition :apigateway: region::/clientcertificates`
Implementazione	`arn:partition :apigateway: region::/restapis/ api-id/deployments/ id`
Distribuzioni	`arn:partition :apigateway: region::/restapis/ api-id/deployments`
DocumentationPart	`arn:partition :apigateway: region::/restapis/ api-id/documentation/parts/ id`
DocumentationParts	`arn:partition :apigateway: region::/restapis/ api-id/documentation/parts`

Risorsa	ARN
DocumentationVersion	`arn:partition :apigateway: region::/restapis/ api-id/documentation/versions/ version`
DocumentationVersions	`arn:partition :apigateway: region::/restapis/ api-id/documentation/versions`
DomainName	`arn:partition :apigateway: region::/domainnames/ domain-name`
DomainNames	`arn:partition :apigateway: region::/domainnames`
GatewayResponse	`arn:partition :apigateway: region::/restapis/ api-id/gatewayresponses/ response-type`
GatewayResponses	`arn:partition :apigateway: region::/restapis/ api-id/gatewayresponses`
Integrazione	`arn:partition :apigateway: region::/restapis/ api-id/resources/ resource-id /methods/ http-method /integration`
IntegrationResponse	`arn:partition :apigateway: region::/restapis/ api-id/resources/ resource-id /methods/ http-method /integration/responses/ status-code`

Risorsa	ARN
Metodo	`arn:partition :apigateway: region::/restapis/ api-id/resources/ resource-id /methods/ http-method`
MethodResponse	`arn:partition :apigateway: region::/restapis/ api-id/resources/ resource-id /methods/ http-method /responses/ status-code`
Modello	`arn:partition :apigateway: region::/restapis/ api-id/models/model-name`
Modelli	`arn:partition :apigateway: region::/restapis/ api-id/models`
RequestValidator	`arn:partition :apigateway: region::/restapis/ api-id/requestvalidators/ id`
RequestValidators	`arn:partition :apigateway: region::/restapis/ api-id/requestvalidators`
Risorsa	`arn:partition :apigateway: region::/restapis/ api-id/resources/ id`
Risorse	`arn:partition :apigateway: region::/restapis/ api-id/resources`

Risorsa	ARN
RestApi	`arn:partition :apigateway: region::/restapis/ api-id`
RestApis	`arn:partition :apigateway: region::/restapis`
Stage	`arn:partition :apigateway: region::/restapis/ api-id/stages/stage-name`
Stage	`arn:partition :apigateway: region::/restapis/ api-id/stages`
Tag	`arn:partition :apigateway: region::/tags/url-encoded-resource-arn`
Modello	`arn:partition :apigateway: region::/restapis/models/ model-name /template`
UsagePlan	`arn:partition :apigateway: region::/usageplans/ usageplan-id`
UsagePlans	`arn:partition :apigateway: region::/usageplans`
UsagePlanKey	`arn:partition :apigateway: region::/usageplans/ usageplan-id /keys/id`
UsagePlanKeys	`arn:partition :apigateway: region::/usageplans/ usageplan-id /keys`

Risorsa	ARN
VpcLink	`arn:partition :apigateway: region::/vpclinks/ vpclink-id`
VpcLinks	`arn:partition :apigateway: region::/vpclinks`

`execute-api`(API HTTP, WebSocket API e API REST)

Risorsa	ARN
WebSocket Endpoint API	`arn:partition :execute-api: region:account-id :api-id/stage/route-key`
Endpoint API HTTP e REST API *	`arn:partition :execute-api: region:account-id :api-id/stage/http-method /resource-path`
Provider di autorizzazioni Lambda **	`arn:partition :execute-api: region:account-id :api-id/authorizers/ authorizer-id`

* L'ARN dell'endpoint di instradamento `$default` per le API HTTP è `arn:partition:execute-api:region:account-id:api-id/*/$default`.

** Questo ARN è applicabile solo quando si imposta la condizione `SourceArn` nella policy delle risorse per una funzione Lambda di autorizzazione. Per un esempio, consulta the section called “Creare un'autorizzazione Lambda”.

APIEstensioni aperte per API Gateway

Le estensioni API Gateway supportano l'autorizzazione AWS specifica e le integrazioni API specifiche per Gateway API per e. REST APIs HTTP APIs In questa sezione, descriviamo le estensioni API Gateway secondo la specifica Open. API

> **Tip**
>
> Per comprendere come vengono utilizzate le estensioni API Gateway in un'applicazione, è possibile utilizzare la console API Gateway per creare un REST API or HTTP API ed esportarlo in un file di API definizione aperto. Per ulteriori informazioni su come esportare un fileAPI, vedere Esporta un file REST API da API Gateway eEsportazione di API HTTP da API Gateway.

Argomenti

- x-amazon-apigateway-anyoggetto -method
- x-amazon-apigateway-cors oggetto
- x-amazon-apigateway-apiproprietà -key-source
- x-amazon-apigateway-auth oggetto
- x-amazon-apigateway-authorizer oggetto
- x-amazon-apigateway-authtype proprietà
- x-amazon-apigateway-binaryproprietà -media-types
- x-amazon-apigateway-documentation oggetto
- x-amazon-apigateway-endpoint-oggetto di configurazione
- x-amazon-apigateway-gatewayoggetto -response
- x-amazon-apigateway-gateway-risposte. gatewayResponse oggetto
- x-amazon-apigateway-gateway-risposte. responseParameters oggetto
- x-amazon-apigateway-gateway-risposte. responseTemplates oggetto
- x-amazon-apigateway-importexport-versione
- x-amazon-apigateway-integration oggetto
- x-amazon-apigateway-integrations oggetto
- x-amazon-apigateway-integration. requestTemplates oggetto

- x-amazon-apigateway-integration. requestParameters oggetto
- x-amazon-apigateway-integrationoggetto.response
- x-amazon-apigateway-integrationoggetto.response
- x-amazon-apigateway-integration. responseTemplates oggetto
- x-amazon-apigateway-integration. responseParameters oggetto
- x-amazon-apigateway-integration. tlsConfig oggetto
- x-amazon-apigateway-minimum-dimensione di compressione
- x-amazon-apigateway-policy
- x-amazon-apigateway-requestproprietà -validator
- x-amazon-apigateway-requestoggetto -validators
- x-amazon-apigateway-request-validatori. requestValidator oggetto
- x-amazon-apigateway-tagproprietà -value

x-amazon-apigateway-anyoggetto -method

Speciifica l'oggetto Open API Operation per il `ANY` metodo catch-all API Gateway in un oggetto Open API Path Item. Questo oggetto può esistere insieme ad altri oggetti Operation e catturerà qualsiasi HTTP metodo che non sia stato dichiarato esplicitamente.

La tabella seguente elenca le proprietà estese da API Gateway. Per le altre proprietà Open API Operation, vedere la API specifica Open.

Nome proprietà	Tipo	Descrizione
`isDefaultRoute`	`Boolean`	Specificare se una route è la route `$default`. Supportat o solo per HTTPAPIs. Per ulteriori informazioni, consulta Crea percorsi per le API HTTP in API Gateway.
`x-amazon-apigateway-integration`	x-amazon-apigateway-integra tion oggetto	Specifica l'integrazione del metodo con il back-end. Si tratta di una proprietà estesa dell'oggetto Open

Nome proprietà	Tipo	Descrizione
		API Operation. L'integrazione può essere del tipo `AWS`, `AWS_PROXY` , `HTTP`, `HTTP_PROXY` o `MOCK`.

x-amazon-apigateway-any-method: esempi

L'esempio seguente integra il metodo `ANY` per una risorsa proxy, `{proxy+}`, con una funzione Lambda, `TestSimpleProxy`.

```
    "/{proxy+}": {
      "x-amazon-apigateway-any-method": {
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "proxy",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {},
        "x-amazon-apigateway-integration": {
          "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789012:function:TestSimpleProxy/invocations",
          "httpMethod": "POST",
          "type": "aws_proxy"
        }
```

L'esempio seguente crea un `$default` percorso per an HTTP API che si integra con una funzione Lambda,. `HelloWorld`

```
"/$default": {
    "x-amazon-apigateway-any-method": {
      "isDefaultRoute": true,
      "x-amazon-apigateway-integration": {
```

```
        "type": "AWS_PROXY",
        "httpMethod": "POST",
        "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789012:function:HelloWorld/invocations",
        "timeoutInMillis": 1000,
        "connectionType": "INTERNET",
        "payloadFormatVersion": 1.0
      }
   }
}
```

x-amazon-apigateway-cors oggetto

Specifica la configurazione di condivisione delle risorse tra le origini (CORS) per un. HTTP API L'estensione si applica alla struttura Open a livello di root. API Per ulteriori informazioni, consulta Configurazione CORS per HTTP APIs in API Gateway.

Nome proprietà	Tipo	Descrizione
`allowOrigins`	`Array`	Specifica le origini consentite.
`allowCredentials`	`Boolean`	Speciifica se le credenziali sono incluse nella richiesta. CORS
`exposeHeaders`	`Array`	Specifica le intestazioni esposte.
`maxAge`	`Integer`	Specifica il numero di secondi in cui il browser deve memorizzare nella cache i risultati delle richieste preliminari.
`allowMethods`	`Array`	Speciifica i metodi consentiti. HTTP
`allowHeaders`	`Array`	Specifica le intestazioni consentite.

x-amazon-apigateway-cors esempio

Di seguito è riportato un esempio CORS di configurazione per un HTTPAPI.

```
"x-amazon-apigateway-cors": {
    "allowOrigins": [
      "https://www.example.com"
    ],
    "allowCredentials": true,
    "exposeHeaders": [
      "x-apigateway-header",
      "x-amz-date",
      "content-type"
    ],
    "maxAge": 3600,
    "allowMethods": [
      "GET",
      "OPTIONS",
      "POST"
    ],
    "allowHeaders": [
      "x-apigateway-header",
      "x-amz-date",
      "content-type"
    ]
}
```

x-amazon-apigateway-apiproprietà -key-source

Specificate la fonte che riceverà una API chiave per limitare i API metodi che richiedono una chiave. Questa proprietà API di livello è un tipo. `String` Per ulteriori informazioni sulla configurazione di un metodo per richiedere una API chiave, vedere. the section called “Configura un metodo per utilizzare API le chiavi con una definizione aperta API”

Specificare l'origine della API chiave per le richieste. I valori validi sono:

- `HEADER`per ricevere la API chiave dall'`X-API-Key`intestazione di una richiesta.
- `AUTHORIZER`per ricevere la API chiave `UsageIdentifierKey` da un autorizzatore Lambda (precedentemente noto come autorizzatore personalizzato).

x-amazon-apigateway-apiesempio di -key-source

L'esempio seguente imposta l'`X-API-Key`intestazione come fonte chiave. API

OpenAPI 2.0

```
{
  "swagger" : "2.0",
  "info" : {
    "title" : "Test1"
   },
  "schemes" : [ "https" ],
  "basePath" : "/import",
  "x-amazon-apigateway-api-key-source" : "HEADER",
   .
   .
   .
}
```

OpenAPI 3.0.1

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "Test1"
  },
  "servers" : [ {
    "url" : "/{basePath}",
    "variables" : {
      "basePath" : {
        "default" : "import"
      }
    }
  } ],
  "x-amazon-apigateway-api-key-source" : "HEADER",
   .
   .
   .
}
```

x-amazon-apigateway-auth oggetto

Definisce un tipo di autorizzazione da applicare per l'autorizzazione delle chiamate ai metodi in API Gateway.

Nome proprietà	Tipo	Descrizione
`type`	`string`	Specifica il tipo di autorizzazione. Specifica `"NONE"` per l'accesso aperto. Specifica `"AWS_IAM"` per l'utilizzo di autorizzazioni IAM. I valori non rispettano la distinzione tra maiuscole e minuscole.

x-amazon-apigateway-auth esempio

L'esempio seguente imposta il tipo di autorizzazione per un API metodo.

OpenAPI 3.0.1

```
{
  "openapi": "3.0.1",
  "info": {
    "title": "openapi3",
    "version": "1.0"
  },
  "paths": {
    "/protected-by-iam": {
      "get": {
        "x-amazon-apigateway-auth": {
          "type": "AWS_IAM"
        }
      }
    }
  }
}
```

x-amazon-apigateway-authorizer oggetto

Definisce un autorizzatore Lambda, un pool di utenti Amazon Cognito o un JWT autorizzatore da applicare per l'autorizzazione delle chiamate ai metodi in Gateway. API Questa estensione si applica alla definizione di sicurezza in Open 2 e Open 3. API API

Nome proprietà	Tipo	Descrizione
`type`	`string`	Il tipo di autorizzazione. Questa proprietà è obbligato ria. Per RESTAPIs, specifica re `token` un autorizzatore con l'identità del chiamante incorporata in un token di autorizzazione. Specifica re `request` per un'autori zzazione con l'identità dell'inte rmediario contenuta nei parametri della richiesta . Specificare `cognito_u ser_pools` un autorizza tore che utilizzi un pool di utenti Amazon Cognito per controllare l'accesso al tuo. API Per HTTPAPIs, specifica re `request` un autorizza tore Lambda con l'identit à del chiamante contenuta nei parametri della richiesta. Specificare un `jwt` autorizza tore. JWT
`authorizerUri`	`string`	L'Uniform Resource Identifier (URI) della funzione Lambda

Nome proprietà	Tipo	Descrizione
		dell'autorizzatore. La sintassi è esposta di seguito: `"arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/arn:aws:lambda:us-east-1: account-id :function: auth_function_name /invocations"`
`authorizerCredentials`	`string`	Le credenziali richieste per richiamare l'autorizzatore, se del caso, sotto forma di ruolo di esecuzione. ARN IAM Ad esempio, «arn:aws:iam::*account-id* :*IAM_role*".
`authorizerPayloadFormatVersion`	`string`	For HTTPAPIs, specifica il formato dei dati che API Gateway invia a un sistema di autorizzazione Lambda e il API modo in cui Gateway interpreta la risposta di Lambda. Per ulteriori informazioni, consulta the section called "Tipo di formato payload".

Nome proprietà	Tipo	Descrizione
`enableSimpleRespon ses`	`Boolean`	Per HTTPAPIs, specifica se un `request` autorizzatore restituisce un valore booleano o una politica. IAM Supportato solo per le autorizzazioni con un `authorizerPayloadF ormatVersion 2.0`. Se abilitata, la funzione del provider di autorizza zioni Lambda restituisce un valore booleano. Per ulteriori informazioni, consulta the section called “Risposta della funzione Lambda per il formato 2.0”.
`identitySource`	`string`	Un elenco separato da virgole di espressioni di mappatura dei parametri di richiesta come origine di identità. Applicabile solo per l'autorizzazione del tipo `request` e `jwt`.
`jwtConfiguration`	`Object`	Speciifica l'emittente e i destinatari di un autorizzatore. JWT Per ulteriori informazi oni, consulta il riferimento alla versione 2 JWTConfigurationdi API Gateway. API Supportato solo per HTTPAPIs.

Nome proprietà	Tipo	Descrizione
`identityValidationExpression`	`string`	Espressione regolare per la convalida del token come identità in ingresso. Ad esempio, "^x-[a-z]+". Supportato solo per gli `TOKEN` autorizzatori di RESTAPIs.
`authorizerResultTtlInSeconds`	`string`	Numero di secondi durante i quali il risultato dell'auto rizzazione viene memorizzato nella cache.
`providerARNs`	Una matrice di `string`	Un elenco del pool di utenti di Amazon Cognito ARNs per. `COGNITO_USER_POOLS`

x-amazon-apigateway-authorizer esempi per REST APIs

Il seguente esempio API di definizioni di sicurezza Open specifica un autorizzatore Lambda del tipo «token» e denominato. `test-authorizer`

```
 "securityDefinitions" : {
   "test-authorizer" : {
     "type" : "apiKey",                          // Required and the value must be
"apiKey" for an API Gateway API.
     "name" : "Authorization",                   // The name of the header containing
the authorization token.
     "in" : "header",                            // Required and the value must be
"header" for an API Gateway API.
     "x-amazon-apigateway-authtype" : "custom", // Specifies the authorization
mechanism for the client.
     "x-amazon-apigateway-authorizer" : {        // An API Gateway Lambda authorizer
definition
       "type" : "token",                         // Required property and the value
must "token"
```

```
        "authorizerUri" : "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:account-id:function:function-name/invocations",
        "authorizerCredentials" : "arn:aws:iam::account-id:role",
        "identityValidationExpression" : "^x-[a-z]+",
        "authorizerResultTtlInSeconds" : 60
      }
    }
  }
```

Il seguente frammento di oggetto Open API operation imposta l'utilizzo dell'`GET /http`autorizzatore Lambda precedente.

```
   "/http" : {
      "get" : {
        "responses" : { },
        "security" : [ {
          "test-authorizer" : [ ]
        } ],
        "x-amazon-apigateway-integration" : {
          "type" : "http",
          "responses" : {
            "default" : {
              "statusCode" : "200"
            }
          },
          "httpMethod" : "GET",
          "uri" : "http://api.example.com"
        }
      }
    }
```

Il seguente esempio API di definizioni di sicurezza Open specifica un autorizzatore Lambda del tipo «request», con un singolo parametro di intestazione `auth` () come origine dell'identità. Il nome di `securityDefinitions` è `request_authorizer_single_header`.

```
"securityDefinitions": {
    "request_authorizer_single_header" : {
      "type" : "apiKey",
      "name" : "auth",              // The name of a single header or query parameter
 as the identity source.
```

```
      "in" : "header",               // The location of the single identity source
 request parameter. The valid value is "header" or "query"
      "x-amazon-apigateway-authtype" : "custom",
      "x-amazon-apigateway-authorizer" : {
        "type" : "request",
        "identitySource" : "method.request.header.auth",   // Request parameter mapping
 expression of the identity source. In this example, it is the 'auth' header.
        "authorizerCredentials" : "arn:aws:iam::123456789012:role/AWSepIntegTest-CS-
LambdaRole",
        "authorizerUri" : "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:123456789012:function:APIGateway-Request-
Authorizer:vtwo/invocations",
        "authorizerResultTtlInSeconds" : 300
      }
    }
}
```

Il seguente esempio API di definizioni di sicurezza Open specifica un autorizzatore Lambda del tipo «request», con un parametro header `HeaderAuth1` () e un `QueryString1` parametro di stringa di query come fonti di identità.

```
"securityDefinitions": {
    "request_authorizer_header_query" : {
      "type" : "apiKey",
      "name" : "Unused",             // Must be "Unused" for multiple identity sources
 or non header or query type of request parameters.
      "in" : "header",               // Must be "header" for multiple identity sources
 or non header or query type of request parameters.
      "x-amazon-apigateway-authtype" : "custom",
      "x-amazon-apigateway-authorizer" : {
        "type" : "request",
        "identitySource" : "method.request.header.HeaderAuth1,
 method.request.querystring.QueryString1",   // Request parameter mapping expressions
 of the identity sources.
        "authorizerCredentials" : "arn:aws:iam::123456789012:role/AWSepIntegTest-CS-
LambdaRole",
        "authorizerUri" : "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:123456789012:function:APIGateway-Request-
Authorizer:vtwo/invocations",
        "authorizerResultTtlInSeconds" : 300
      }
    }
```

```
}
```

Il seguente esempio API di definizioni di sicurezza Open specifica un autorizzatore API Gateway Lambda del tipo «request», con una variabile a fase singola `stage` () come origine dell'identità.

```
"securityDefinitions": {
    "request_authorizer_single_stagevar" : {
      "type" : "apiKey",
      "name" : "Unused",             // Must be "Unused", for multiple identity sources
 or non header or query type of request parameters.
      "in" : "header",               // Must be "header", for multiple identity sources
 or non header or query type of request parameters.
      "x-amazon-apigateway-authtype" : "custom",
      "x-amazon-apigateway-authorizer" : {
        "type" : "request",
        "identitySource" : "stageVariables.stage",   // Request parameter mapping
 expression of the identity source. In this example, it is the stage variable.
        "authorizerCredentials" : "arn:aws:iam::123456789012:role/AWSepIntegTest-CS-
LambdaRole",
        "authorizerUri" : "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:123456789012:function:APIGateway-Request-
Authorizer:vtwo/invocations",
        "authorizerResultTtlInSeconds" : 300
      }
    }
}
```

Il seguente esempio API di definizione di sicurezza Open specifica un pool di utenti Amazon Cognito come autorizzatore.

```
 "securityDefinitions": {
    "cognito-pool": {
      "type": "apiKey",
      "name": "Authorization",
      "in": "header",
      "x-amazon-apigateway-authtype": "cognito_user_pools",
      "x-amazon-apigateway-authorizer": {
        "type": "cognito_user_pools",
        "providerARNs": [
          "arn:aws:cognito-idp:us-east-1:123456789012:userpool/us-east-1_ABC123"
        ]
      }
```

```
    }
```

Il seguente frammento di oggetto Open API operation imposta l'utilizzo del `GET /http` precedente pool di utenti di Amazon Cognito come autorizzatore, senza ambiti personalizzati.

```
   "/http" : {
      "get" : {
        "responses" : { },
        "security" : [ {
          "cognito-pool" : [ ]
        } ],
        "x-amazon-apigateway-integration" : {
          "type" : "http",
          "responses" : {
            "default" : {
              "statusCode" : "200"
            }
          },
          "httpMethod" : "GET",
          "uri" : "http://api.example.com"
        }
      }
    }
```

x-amazon-apigateway-authorizer esempi per HTTP APIs

Il seguente esempio di Open API 3.0 crea un JWT autorizzatore per un utente HTTP API che utilizza Amazon Cognito come provider di identità, con `Authorization` l'intestazione come fonte di identità.

```
"securitySchemes": {
  "jwt-authorizer-oauth": {
    "type": "oauth2",
     "x-amazon-apigateway-authorizer": {
       "type": "jwt",
       "jwtConfiguration": {
          "issuer": "https://cognito-idp.region.amazonaws.com/userPoolId",
          "audience": [
            "audience1",
            "audience2"
          ]
        },
```

```
        "identitySource": "$request.header.Authorization"
    }
  }
}
```

Il seguente esempio Open API 3.0 produce lo stesso JWT autorizzatore dell'esempio precedente. Tuttavia, questo esempio utilizza la API `openIdConnectUrl` proprietà Open per rilevare automaticamente l'emittente. Il `openIdConnectUrl` deve essere completamente formato.

```
"securitySchemes": {
  "jwt-authorizer-autofind": {
    "type": "openIdConnect",
    "openIdConnectUrl": "https://cognito-idp.region.amazonaws.com/userPoolId/.well-
known/openid-configuration",
    "x-amazon-apigateway-authorizer": {
      "type": "jwt",
      "jwtConfiguration": {
        "audience": [
          "audience1",
          "audience2"
        ]
      },
      "identitySource": "$request.header.Authorization"
    }
  }
}
```

L'esempio seguente crea un autorizzatore Lambda per un. HTTP API In questo esempio l'autorizzazione utilizza l'intestazione `Authorization` come origine di identità. L'autorizzazione utilizza il tipo di formato payload `2.0` e restituisce il valore booleano, poiché `enableSimpleResponses` è impostato su `true`.

```
"securitySchemes" : {
  "lambda-authorizer" : {
    "type" : "apiKey",
    "name" : "Authorization",
    "in" : "header",
    "x-amazon-apigateway-authorizer" : {
      "type" : "request",
      "identitySource" : "$request.header.Authorization",
      "authorizerUri" : "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:function-name/invocations",
```

```
      "authorizerPayloadFormatVersion" : "2.0",
      "authorizerResultTtlInSeconds" : 300,
      "enableSimpleResponses" : true
    }
  }
}
```

x-amazon-apigateway-authtype proprietà

Infatti RESTAPIs, questa estensione può essere utilizzata per definire un tipo personalizzato di autorizzatore Lambda. In questo caso, il valore è in formato libero. Ad esempio, un API può avere più autorizzatori Lambda che utilizzano schemi interni diversi. È possibile utilizzare questa estensione per identificare lo schema interno di un provider di autorizzazioni Lambda.

Più comunemente, in HTTP APIs and RESTAPIs, può anche essere usato per definire IAM l'autorizzazione per diverse operazioni che condividono lo stesso schema di sicurezza. In questo caso, il termine `awsSigv4` è un termine riservato, insieme a qualsiasi termine con prefisso `aws`.

Questa estensione si applica allo schema `apiKey` di sicurezza dei tipi in Open API 2 e Open API 3.

x-amazon-apigateway-authtype esempio

Il seguente esempio di Open API 3 definisce IAM l'autorizzazione su più risorse in una REST API o HTTPAPI:

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "openapi3",
    "version" : "1.0"
  },
  "paths" : {
    "/operation1" : {
      "get" : {
        "responses" : {
          "default" : {
            "description" : "Default response"
          }
        },
        "security" : [ {
          "sigv4Reference" : [ ]
```

```
        } ]
      }
    },
    "/operation2" : {
      "get" : {
        "responses" : {
          "default" : {
            "description" : "Default response"
          }
        },
        "security" : [ {
          "sigv4Reference" : [ ]
        } ]
      }
    }
  },
  "components" : {
    "securitySchemes" : {
      "sigv4Reference" : {
        "type" : "apiKey",
        "name" : "Authorization",
        "in" : "header",
        "x-amazon-apigateway-authtype": "awsSigv4"
      }
    }
  }
}
```

Il seguente esempio di Open API 3 definisce un autorizzatore Lambda con uno schema personalizzato per: REST API

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "openapi3 for REST API",
    "version" : "1.0"
  },
  "paths" : {
    "/protected-by-lambda-authorizer" : {
      "get" : {
        "responses" : {
          "200" : {
            "description" : "Default response"
```

```
          }
        },
        "security" : [ {
          "myAuthorizer" : [ ]
        } ]
      }
    }
  },
  "components" : {
    "securitySchemes" : {
      "myAuthorizer" : {
        "type" : "apiKey",
        "name" : "Authorization",
        "in" : "header",
        "x-amazon-apigateway-authorizer" : {
          "identitySource" : "method.request.header.Authorization",
          "authorizerUri" : "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:account-id:function:function-name/invocations",
          "authorizerResultTtlInSeconds" : 300,
          "type" : "request",
          "enableSimpleResponses" : false
        },
        "x-amazon-apigateway-authtype": "Custom scheme with corporate claims"
      }
    }
  },
  "x-amazon-apigateway-importexport-version" : "1.0"
}
```

Consulta anche

autorizzatore. authType

x-amazon-apigateway-binaryproprietà -media-types

Speciifica l'elenco dei tipi di file multimediali binari che devono essere supportati da API Gateway, ad esempio e. `application/octet-stream` `image/jpeg` Questa estensione è una JSON matrice. Dovrebbe essere inclusa come estensione di primo livello fornita dal fornitore del documento OpenAPI.

x-amazon-apigateway-binaryesempio di -media-types

L'esempio seguente mostra l'ordine di ricerca della codifica di un. API

```
"x-amazon-apigateway-binary-media-types": [ "application/octet", "image/jpeg" ]
```

x-amazon-apigateway-documentation oggetto

Definisce le parti della documentazione da importare in API Gateway. Questo oggetto è un JSON oggetto contenente una matrice di `DocumentationPart` istanze.

Nome proprietà	Tipo	Descrizione
`documentationParts`	`Array`	Matrice delle istanze di `DocumentationPart` esportate o importate.
`version`	`String`	L'identificatore di versione dello snapshot delle parti di documentazione esportate.

x-amazon-apigateway-documentation esempio

L'esempio seguente dell'estensione API Gateway a Open API definisce `DocumentationParts` le istanze da importare o esportare da un Gateway API inAPI.

```
{ ...
  "x-amazon-apigateway-documentation": {
    "version": "1.0.3",
    "documentationParts": [
       {
         "location": {
           "type": "API"
       },
        "properties": {
          "description": "API description",
          "info": {
            "description": "API info description 4",
            "version": "API info version 3"
```

```
          }
        }
      },
      {
         … // Another DocumentationPart instance
      }
    ]
  }
}
```

x-amazon-apigateway-endpoint-oggetto di configurazione

Specifica i dettagli della configurazione dell'endpoint per un. API Questa estensione è una proprietà estesa dell'oggetto Open API Operation. Questo oggetto deve essere presente nelle estensioni dei fornitori di primo livello per Swagger 2.0. Per Open API 3.0, dovrebbe essere presente nelle estensioni del fornitore dell'oggetto Server.

Nome proprietà	Tipo	Descrizione
`disableExecuteApiEndpoint`	Booleano	Speciifica se i client possono richiamare il tuo API utilizzando l'endpoint predefinito. `execute-api` Per impostazione predefinita, i client possono richiamare il tuo API con l'endpoint predefinito. `https://{api_id}.execute-api.{region}.amazonaws.com` Per richiedere che i client utilizzino un nome di dominio personalizzato per richiamare il tuoAPI, specifica. `true`
`vpcEndpointIds`	Una matrice di `String`	Un elenco di VpcEndpoint identificatori in base ai quali creare record di alias Route 53 per un. REST API È supportat

Nome proprietà	Tipo	Descrizione
		o solo per REST APIs il tipo di `PRIVATE` endpoint.

x-amazon-apigateway-endpoint-esempi di configurazione

L'esempio seguente associa gli VPC endpoint specificati a. REST API

```
"x-amazon-apigateway-endpoint-configuration": {
    "vpcEndpointIds": ["vpce-0212a4ababd5b8c3e", "vpce-01d622316a7df47f9"]
}
```

L'esempio seguente disabilita l'endpoint predefinito per un. API

```
"x-amazon-apigateway-endpoint-configuration": {
    "disableExecuteApiEndpoint": true
}
```

x-amazon-apigateway-gatewayoggetto -response

Definisce le risposte del gateway per an API come GatewayResponsemappa stringa-valore di coppie chiave-valore. L'estensione si applica alla struttura Open a livello di root. API

Nome proprietà	Tipo	Descrizione
`responseType`	x-amazon-apigateway-gateway-risposte. gatewayResponse	A `GatewayResponse` per quanto specificato *`responseType`* .

x-amazon-apigateway-gateway-esempio di risposte

L'APIesempio seguente di estensione API Gateway a Open definisce una GatewayResponsesmappa che contiene due GatewayResponseistanze, una per il `DEFAULT_4XX` tipo e l'altra per il tipo. `INVALID_API_KEY`

```
{
  "x-amazon-apigateway-gateway-responses": {
    "DEFAULT_4XX": {
      "responseParameters": {
        "gatewayresponse.header.Access-Control-Allow-Origin": "'domain.com'"
      },
      "responseTemplates": {
        "application/json": "{\"message\": test 4xx b }"
      }
    },
    "INVALID_API_KEY": {
      "statusCode": "429",
      "responseTemplates": {
        "application/json": "{\"message\": test forbidden }"
      }
    }
  }
}
```

x-amazon-apigateway-gateway-risposte. gatewayResponse oggetto

Definisce una risposta del gateway di un tipo di risposta specificato, inclusi il codice di stato, eventuali modelli di risposta o parametri di risposta applicabili.

Nome proprietà	Tipo	Descrizione
responseParameters	x-amazon-apigateway-gateway-risposte. responseParameters	Specificate i GatewayResponseparametri, vale a dire i parametri dell'intestazione. I valori di parametro possono assumere qualsiasi valore di parametro di richiesta in ingresso o un valore personalizzato statico.
responseTemplates	x-amazon-apigateway-gateway-risposte. responseTemplates	Specifica i modelli di mappatura della risposta del gateway. I modelli non

Nome proprietà	Tipo	Descrizione
		vengono elaborati dal VTL motore.
statusCode	`string`	Un codice di HTTP stato per la risposta del gateway.

x-amazon-apigateway-gateway-risposte. gatewayResponse esempio

L'esempio seguente dell'estensione API Gateway di Open API definisce un GatewayResponsemodo per personalizzare la `INVALID_API_KEY` risposta in modo da restituire il codice di stato`456`, il valore dell'`api-key`intestazione della richiesta in entrata e un `"Bad api-key"` messaggio.

```
"INVALID_API_KEY": {
  "statusCode": "456",
  "responseParameters": {
    "gatewayresponse.header.api-key": "method.request.header.api-key"
  },
  "responseTemplates": {
    "application/json": "{\"message\": \"Bad api-key\" }"
  }
}
```

x-amazon-apigateway-gateway-risposte. responseParameters oggetto

Definisce una string-to-string mappa di coppie chiave-valore per generare i parametri di risposta del gateway dai parametri della richiesta in entrata o utilizzando stringhe letterali. Supportato solo per. REST APIs

Nome proprietà	Tipo	Descrizione
`gatewayresponse.` *param-position* `.` *param-name*	`string`	*param-position* può essere `header`, `path` o `querystring` . Per ulteriori

Nome proprietà	Tipo	Descrizione
		informazioni, consulta Come mappare i dati di richiesta di metodi ai parametri di richiesta di integrazione.

x-amazon-apigateway-gateway-risposte. responseParameters esempio

Il seguente esempio di API estensioni Open mostra un'espressione di mappatura dei parametri di GatewayResponserisposta per abilitare CORS il supporto per le risorse nei `*.example.domain` domini.

```
"responseParameters": {
  "gatewayresponse.header.Access-Control-Allow-Origin": '*.example.domain',
  "gatewayresponse.header.from-request-header" : method.request.header.Accept,
  "gatewayresponse.header.from-request-path" : method.request.path.petId,
  "gatewayresponse.header.from-request-query" : method.request.querystring.qname
}
```

x-amazon-apigateway-gateway-risposte. responseTemplates oggetto

Definisce i modelli di GatewayResponsemappatura, come una string-to-string mappa di coppie chiave-valore, per una determinata risposta del gateway. Per ogni coppia chiave-valore, la chiave è il tipo di contenuto. Ad esempio, «application/json» e il valore è un modello di mappatura stringified per semplici sostituzioni variabili. Un modello di `GatewayResponse` mappatura non viene elaborato dal motore Velocity Template Language (). VTL

Nome proprietà	Tipo	Descrizione
`content-type`	`string`	Modello di mappatura del corpo `GatewayResponse` che supporta solo la semplice sostituzione di variabili per

Nome proprietà	Tipo	Descrizione
		personalizzare un corpo della risposta del gateway.

x-amazon-apigateway-gateway-risposte. responseTemplates esempio

Il seguente esempio di API estensioni Open mostra un modello di GatewayResponsemappatura per personalizzare una risposta di errore API generata dal gateway in un formato specifico dell'app.

```
      "responseTemplates": {
        "application/json": "{ \"message\": $context.error.messageString, \"type\": 
$context.error.responseType, \"statusCode\": '488' }"
      }
```

Il seguente esempio di API estensioni Open mostra un modello di GatewayResponsemappatura per sostituire una risposta di errore generata dal API Gateway con un messaggio di errore statico.

```
      "responseTemplates": {
        "application/json": "{ \"message\": 'API-specific errors' }"
      }
```

x-amazon-apigateway-importexport-versione

Speciifica la versione dell'algoritmo di importazione ed esportazione API Gateway per. HTTP APIs Attualmente, l'unico valore supportato è `1.0`. Per ulteriori informazioni, consulta exportVersionla sezione APIGateway Version 2 API Reference.

x-amazon-apigateway-importexport-esempio di versione

Nell'esempio seguente viene impostata la versione di importazione ed esportazione su `1.0`.

```
{
    "openapi": "3.0.1",
    "x-amazon-apigateway-importexport-version": "1.0",
    "info": { ...
```

x-amazon-apigateway-integration oggetto

Specifica i dettagli dell'integrazione di back-end utilizzati per questo metodo. Questa estensione è una proprietà estesa dell'oggetto Open API Operation. Il risultato è un oggetto di integrazione API Gateway.

Nome proprietà	Tipo	Descrizione
`cacheKeyParameters`	Una matrice di `string`	Elenco di parametri di richiesta i cui valori devono essere memorizzati nella cache.
`cacheNamespace`	`string`	Un gruppo API di tag specifico di parametri correlati memorizzati nella cache.
`connectionId`	`string`	L'ID di a VpcLinkper l'integra zione privata.
`connectionType`	`string`	Il tipo di connessione dell'inte grazione. Il valore valido è `"VPC_LINK"` per l'integra zione privata, in caso contrario , `"INTERNET"` .
`credentials`	`string`	Per le credenziali AWS IAM basate sui ruoli, specifica il ruolo ARN appropriato. IAM Se non specificato, per impostazione predefinita, le credenziali sono autorizza zioni basate sulle risorse che devono essere aggiunte manualmente per consentire loro di accedere alla risorsa. API Per ulteriori informazioni, consulta l'argomento relativo alla concessione di autorizza

Nome proprietà	Tipo	Descrizione
		zioni mediante una policy per le risorse. Nota: quando usi IAM le credenziali, assicurati che gli endpoint AWS STS regionali siano abilitati per la regione in cui vengono distribuite per ottenere prestazioni ottimali. API
`contentHandling`	`string`	Tipi di conversione per la codifica del payload delle richieste. Valori validi sono 1) `CONVERT_TO_TEXT` , per la conversione di un payload binario in una stringa con codifica base64 o per la conversione di un payload di testo in una stringa con codifica `utf-8` o per passare il payload di testo in modo nativo senza modifica e 2) `CONVERT_TO_BINARY` , per la conversione di un payload di testo nel BLOB con decodifica base64 o per passare un payload binario in modo nativo senza modifica.
`httpMethod`	`string`	Il HTTP metodo utilizzato nella richiesta di integrazione. Per le invocazioni della funzione Lambda, il valore deve essere `POST`.

Nome proprietà	Tipo	Descrizione
`integrationSubtype`	`string`	Speciifica il sottotipo di integrazione per l'integra zione di un AWS servizio. Supportato solo per. HTTP APIs Per i sottotipi di integrazi one supportati, consulta the section called “AWS riferimen to alle integrazioni di servizi”.
`passthroughBehavior`	`string`	Specifica il modo in cui deve essere passato un payload di richiesta del tipo di contenuto non mappato attraverso la richiesta di integrazione senza modifiche. I valori supportati sono `when_no_templates` , `when_no_match` e `never`. Per ulteriori informazioni, consulta Integrazione. passthroughBehavior.
`payloadFormatVersion`	`string`	Specifica il formato del payload inviato a un'integr azione. Necessario per HTTPAPIs. Infatti HTTPAPIs, i valori supportati per le integrazioni del proxy Lambda sono e. `1.0` `2.0` Per tutte le altre integrazioni, `1.0` è l'unico valore supportato. Per ulteriori informazioni, consulta the section called “AWS Lambda integrazioni” e the section called “AWS riferimento alle integrazioni di servizi”.

Nome proprietà	Tipo	Descrizione
`requestParameters`	x-amazon-apigateway-integration. requestParameters oggetto	For RESTAPIs, specifica le mappature dai parametri di richiesta del metodo ai parametri della richiesta di integrazione. I parametri di richiesta supportati sono `querystring` , `path`, `header` e `body`. Infatti HTTPAPIs, i parametri di richiesta sono una mappa chiave-valore che specifica i parametri che vengono passati alle integrazioni con un valore specificato. `AWS_PROXY` `integrationSubtype` È possibile fornire valori statici o dati di richiesta di mappatura , variabili di stadio o variabili di contesto che vengono valutate in fase di esecuzione. Per ulteriori informazioni, consulta the section called "AWS integrazioni di servizi".
`requestTemplates`	x-amazon-apigateway-integration. requestTemplates oggetto	Modelli di mappatura per un payload di richieste di tipi specifici. MIME
`responses`	x-amazon-apigateway-integrationoggetto.response	Definisce le risposte di metodo e specifica le mappature di payload o dei parametri desiderate dalle risposte di integrazione alle risposte di metodo.

Nome proprietà	Tipo	Descrizione
`timeoutInMillis`	`integer`	Timeout di integrazione tra 50 ms e 29.000 ms.
`type`	`string`	Il tipo di integrazione con il back-end specificato. I valori validi sono: • `http`oppure`http_proxy`, per l'integrazione con un HTTP backend. • `aws_proxy`, per l'integrazione con le funzioni AWS Lambda. • `aws`, per l'integrazione con funzioni AWS Lambda o altri AWS servizi, come Amazon DynamoDB, Amazon Simple Notification Service o Amazon Simple Queue Service. • `mock`, per l'integrazione con API Gateway senza richiamare alcun backend. Per ulteriori informazioni sui tipi di integrazione, consulta integration:type.
`tlsConfig`	the section called “x-amazon-apigateway-integration.tlsConfig”	Specifica la TLS configurazione per un'integrazione.

Nome proprietà	Tipo	Descrizione
`uri`	`string`	L'endpoint URI del backend. Per integrazioni di questo `aws` tipo, questo è un valore. ARN Per l'HTTPintegrazione, si tratta URL dell'HTTPendpoint che include lo schema `https` or`http`.

x-amazon-apigateway-integration esempi

Infatti HTTPAPIs, puoi definire le integrazioni nella sezione componenti della tua API definizione aperta. Per ulteriori informazioni, consulta x-amazon-apigateway-integrations oggetto.

```
"x-amazon-apigateway-integration": {
    "$ref": "#/components/x-amazon-apigateway-integrations/integration1"
}
```

L'esempio seguente crea un'integrazione con una funzione Lambda. A scopo dimostrativo, si presume che i modelli di mappatura `responseTemplates` di esempio mostrati in `requestTemplates` e negli esempi seguenti si applichino al seguente JSON payload in formato: `{ "name":"value_1", "key":"value_2", "redirect": {"url" :"..."} }` per generare un JSON output di o un output di. `{ "stage":"value_1", "user-id":"value_2" }` XML `<stage>value_1</stage>`

```
"x-amazon-apigateway-integration" : {
    "type" : "aws",
    "uri" : "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:012345678901:function:HelloWorld/invocations",
    "httpMethod" : "POST",
    "credentials" : "arn:aws:iam::012345678901:role/apigateway-invoke-lambda-exec-
role",
    "requestTemplates" : {
                "application/json" : "#set ($root=$input.path('$')) { \"stage\":
 \"$root.name\", \"user-id\": \"$root.key\" }",
                "application/xml" : "#set ($root=$input.path('$')) <stage>$root.name</
stage> "
```

```
    },
    "requestParameters" : {
        "integration.request.path.stage" : "method.request.querystring.version",
        "integration.request.querystring.provider" :
 "method.request.querystring.vendor"
    },
    "cacheNamespace" : "cache namespace",
    "cacheKeyParameters" : [],
    "responses" : {
        "2\\d{2}" : {
            "statusCode" : "200",
            "responseParameters" : {
                "method.response.header.requestId" : "integration.response.header.cid"
            },
            "responseTemplates" : {
                "application/json" : "#set ($root=$input.path('$')) { \"stage\":
 \"$root.name\", \"user-id\": \"$root.key\" }",
                "application/xml" : "#set ($root=$input.path('$')) <stage>$root.name</
stage> "
            }
        },
        "302" : {
            "statusCode" : "302",
            "responseParameters" : {
                "method.response.header.Location" :
 "integration.response.body.redirect.url"
            }
        },
        "default" : {
            "statusCode" : "400",
            "responseParameters" : {
                "method.response.header.test-method-response-header" : "'static value'"
            }
        }
    }
}
```

Nota che le virgolette doppie («) per la JSON stringa nei modelli di mappatura devono essere string-escape (\»).

x-amazon-apigateway-integrations oggetto

Definisce una raccolta di integrazioni. Puoi definire le integrazioni nella sezione componenti della tua API definizione aperta e riutilizzare le integrazioni per più percorsi. Supportato solo per. HTTP APIs

Nome proprietà	Tipo	Descrizione
integration	x-amazon-apigateway-integration oggetto	Una raccolta di oggetti di integrazione.

x-amazon-apigateway-integrations esempio

L'esempio seguente crea un file HTTP API che definisce due integrazioni e fa riferimento alle integrazioni utilizzando. `$ref": "#/components/x-amazon-apigateway-integrations/`*`integration-name`*

```
{
  "openapi": "3.0.1",
  "info":
    {
      "title": "Integrations",
      "description": "An API that reuses integrations",
      "version": "1.0"
    },
  "servers": [
  {
    "url": "https://example.com/{basePath}",
    "description": "The production API server",
    "variables":
      {
        "basePath":
          {
            "default": "example/path"
          }
      }
  }],
  "paths":
    {
      "/":
        {
```

```
          "get":
            {
              "x-amazon-apigateway-integration":
                {
                  "$ref": "#/components/x-amazon-apigateway-integrations/integration1"

                }
            }
        },
       "/pets":
        {
          "get":
            {
              "x-amazon-apigateway-integration":
                {
                  "$ref": "#/components/x-amazon-apigateway-integrations/integration1"

                }
            }
        },
       "/checkout":
        {
          "get":
            {
              "x-amazon-apigateway-integration":
                {
                  "$ref": "#/components/x-amazon-apigateway-integrations/integration2"
                }
            }
        }
    },
    "components": {
      "x-amazon-apigateway-integrations":
        {
          "integration1":
            {
              "type": "aws_proxy",
              "httpMethod": "POST",
              "uri": "arn:aws:apigateway:us-east-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-2:123456789012:function:my-function/invocations",
              "passthroughBehavior": "when_no_templates",
              "payloadFormatVersion": "1.0"
            },
          "integration2":
```

```
            {
              "type": "aws_proxy",
              "httpMethod": "POST",
              "uri": "arn:aws:apigateway:us-east-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-2:123456789012:function:example-function/invocations",
              "passthroughBehavior": "when_no_templates",
              "payloadFormatVersion" : "1.0"
            }
        }
    }
}
```

x-amazon-apigateway-integration. requestTemplates oggetto

Specifica i modelli di mappatura per un payload di richieste dei tipi specificati. MIME

Nome proprietà	Tipo	Descrizione
MIME type	`string`	Un esempio del tipo è. MIME `application/json` Per informazioni su come creare un modello di mappatura, consulta PetStore modello di mappatura.

x-amazon-apigateway-integration. requestTemplates esempio

L'esempio seguente imposta i modelli di mappatura per un payload di richieste di tipi `application/json` and `application/xml`MIME.

```
"requestTemplates" : {
    "application/json" : "#set ($root=$input.path('$')) { \"stage\": \"$root.name\",
 \"user-id\": \"$root.key\" }",
    "application/xml" : "#set ($root=$input.path('$')) <stage>$root.name</stage> "
}
```

x-amazon-apigateway-integration. requestParameters oggetto

Per RESTAPIs, specifica le mappature dai parametri di richiesta del metodo denominato ai parametri della richiesta di integrazione. Prima di fare riferimento ai parametri delle richieste dei metodi, è necessario definirli.

For HTTPAPIs, specifica i parametri che vengono passati alle `AWS_PROXY` integrazioni con un valore specificato. `integrationSubtype`

Nome proprietà	Tipo	Descrizione
`integration.request.`*`<param-type>`*`.`*`<param-name>`	`string`	Perché RESTAPIs, il valore è in genere un parametro di richiesta del metodo predefinito del `method.request.`*`<param-type>`*`.`*`<param-name>` formato, dove `<param-type>` può esserequerystring ,path, header o. body Tuttavia, sono validi anche `$context.`*`VARIABLE_NAME`* , `$stageVariables.`*`VARIABLE_NAME`* e *`STATIC_VALUE`* . Per il body parametro, `<param-name>` è un'espressione di JSON percorso senza il `$.` prefisso.
`parameter`	`string`	Infatti HTTPAPIs, i parametri di richiesta sono una mappa chiave-valore che specifica i parametri che vengono passati alle `AWS_PROXY` integrazioni con un valore specificato.

Nome proprietà	Tipo	Descrizione
		`integrationSubtype` È possibile fornire valori statici o dati di richiesta di mappatura , variabili di stadio o variabili di contesto che vengono valutate in fase di esecuzion e. Per ulteriori informazioni, consulta the section called "AWS integrazioni di servizi".

`x-amazon-apigateway-integration.requestParameters`Esempio

L'esempio di mappatura dei parametri di richiesta seguente converte i parametri di query (`version`), intestazione (`x-user-id`) e percorso (`service`) della richiesta del metodo rispettivamente in parametri di query (`stage`), intestazione (`x-userid`) e percorso (op) della richiesta di integrazione.

> **Note**
>
> Se stai creando risorse tramite Open API or AWS CloudFormation, i valori statici devono essere racchiusi tra virgolette singole.
> Per aggiungere questo valore dalla console, immetti `application/json` nella casella, senza virgolette.

```
"requestParameters" : {
    "integration.request.querystring.stage" : "method.request.querystring.version",
    "integration.request.header.x-userid" : "method.request.header.x-user-id",
    "integration.request.path.op" : "method.request.path.service"
},
```

x-amazon-apigateway-integrationoggetto.response

Definisce le risposte di metodo e specifica le mappature di payload o dei parametri dalle risposte di integrazione alle risposte di metodo.

Nome proprietà	Tipo	Descrizione
`Response status pattern`	x-amazon-apigateway-integrationoggetto.response	Un'espressione regolare utilizzata per abbinare la risposta di integrazione alla risposta del metodo, o `default` per raccogliere qualsiasi risposta che non è stata configurata. Per HTTP le integrazioni, l'espressione regolare si applica al codice di stato della risposta di integrazione. Per le chiamate Lambda, l'espressione regolare si applica al `errorMessage` campo dell'oggetto informativo sull'errore restituito da AWS Lambda come corpo di risposta all'errore quando l'esecuzione della funzione Lambda genera un'eccezione. **Note** Il *`Response status pattern`* il nome della proprietà si riferisce a un codice di stato della risposta o a un'espressione regolare che descrive un gruppo di codici di stato

Nome proprietà	Tipo	Descrizione
		della risposta. Non corrisponde a nessun identificatore di una IntegrationResponserisorsa nel Gateway. API REST API

`x-amazon-apigateway-integration.responses`Esempio

L'esempio seguente mostra un elenco di risposte dalle risposte `2xx` e `302`. Per la `2xx` risposta, la risposta del metodo viene mappata dal payload della risposta di integrazione di tipo or. `application/json` `application/xml` MIME Questa risposta usa i modelli di mappatura forniti. Per la risposta `302`, la risposta del metodo restituisce un'intestazione `Location` il cui valore viene derivato dalla proprietà `redirect.url` nel payload della risposta di integrazione.

```
"responses" : {
    "2\\d{2}" : {
        "statusCode" : "200",
        "responseTemplates" : {
            "application/json" : "#set ($root=$input.path('$')) { \"stage\":
 \"$root.name\", \"user-id\": \"$root.key\" }",
            "application/xml" : "#set ($root=$input.path('$')) <stage>$root.name</
stage> "
        }
    },
    "302" : {
        "statusCode" : "302",
        "responseParameters" : {
            "method.response.header.Location": "integration.response.body.redirect.url"
        }
    }
}
```

x-amazon-apigateway-integrationoggetto.response

Definisce una risposta e specifica le mappature di payload o dei parametri dalla risposta di integrazione alla risposta di metodo.

Nome proprietà	Tipo	Descrizione
`statusCode`	`string`	HTTPcodice di stato per la risposta del metodo; ad esempio,`"200"`. Deve corrispondere a una risposta corrispondente nel `responses` campo APIOperazione aperta.
`responseTemplates`	x-amazon-apigateway-integration. responseTemplates oggetto	Specifica modelli di MIME mappatura specifici del tipo per il payload della risposta.
`responseParameters`	x-amazon-apigateway-integration. responseParameters oggetto	Specifica le mappature dei parametri per la risposta. Solo i parametri `header` e `body` della risposta di integrazione possono essere mappati ai parametri `header` del metodo.
`contentHandling`	`string`	Tipi di conversione per la codifica del payload delle risposte. Valori validi sono 1) `CONVERT_TO_TEXT` , per la conversione di un payload binario in una stringa con codifica base64 o per la conversione di un payload di testo in una stringa con codifica `utf-8` o per passare il payload di testo in modo

Nome proprietà	Tipo	Descrizione
		nativo senza modifica e 2) `CONVERT_TO_BINARY` , per la conversione di un payload di testo nel BLOB con decodifica base64 o per passare un payload binario in modo nativo senza modifica.

`x-amazon-apigateway-integration.response`Esempio

L'esempio seguente definisce una `302` risposta per il metodo che ricava un payload di tipo or dal backend. `application/json` `application/xml` MIME La risposta utilizza i modelli di mappatura forniti e restituisce il reindirizzamento URL dalla risposta di integrazione nell'intestazione del metodo. `Location`

```
{
    "statusCode" : "302",
    "responseTemplates" : {
         "application/json" : "#set ($root=$input.path('$')) { \"stage\": \"$root.name
\", \"user-id\": \"$root.key\" }",
         "application/xml" : "#set ($root=$input.path('$')) <stage>$root.name</stage> "
    },
    "responseParameters" : {
        "method.response.header.Location": "integration.response.body.redirect.url"
    }
}
```

x-amazon-apigateway-integration. responseTemplates oggetto

Specifica i modelli di mappatura per un payload di risposta dei tipi specificati. MIME

Nome proprietà	Tipo	Descrizione
MIME type	string	Specifica un modello di mappatura per trasforma re il corpo della risposta di integrazione nel corpo della risposta del metodo per un determinato tipo. MIME Per informazioni su come creare un modello di mappatura, consulta PetStore modello di mappatura. Un esempio di *MIME type* èapplicati on/json .

x-amazon-apigateway-integration. responseTemplate esempio

L'esempio seguente imposta i modelli di mappatura per un payload di richieste di tipi `application/json` and `application/xml`MIME.

```
"responseTemplates" : {
    "application/json" : "#set ($root=$input.path('$')) { \"stage\": \"$root.name\",
 \"user-id\": \"$root.key\" }",
    "application/xml" : "#set ($root=$input.path('$')) <stage>$root.name</stage> "
}
```

x-amazon-apigateway-integration. responseParameters oggetto

Specifica le mappature dai parametri delle risposte dei metodi di integrazione ai parametri delle risposte di metodo. Puoi mappare `header`, `body` o valori statici al tipo `header` della risposta dei metodi. Supportato solo per RESTAPIs.

Nome proprietà	Tipo	Descrizione
`method.response.header.` *`<param-name>`*	`string`	Il valore di parametro denominato può essere derivato dai tipi `header` e `body` di parametri delle risposte di integrazione.

`x-amazon-apigateway-integration.responseParameters` Esempio

L'esempio seguente mappa i parametri `body` e `header` della risposta di integrazione a due parametri `header` della risposta di metodo.

```
"responseParameters" : {
    "method.response.header.Location" : "integration.response.body.redirect.url",
    "method.response.header.x-user-id" : "integration.response.header.x-userid"
}
```

x-amazon-apigateway-integration. tlsConfig oggetto

Specifica la TLS configurazione per un'integrazione.

Nome proprietà	Tipo	Descrizione
`insecureSkipVerification`	`Boolean`	Supportato solo per RESTAPIs. Speciifica se API Gateway ignora o meno la verifica che il certificato per un endpoint di integrazione sia emesso da un'autorità di certificazione supportata. Questa operazione non è consigliata, ma consente

Nome proprietà	Tipo	Descrizione
		di utilizzare certificati firmati da autorità di certifica zione private o certificati autofirmati. Se abilitato, API Gateway esegue comunque la convalida di base del certifica to, che include il controllo della data di scadenza del certificato, del nome host e della presenza di un'autorità di certificazione principale. Il certificato radice appartene nte all'autorità privata deve soddisfare i seguenti vincoli: • L'estensione x509 `keyUsage` deve avere `keyCertSign` . • L'estensione x509 `basicConstraints` deve avere `CA:TRUE`. Supportato solo per integrazi oni `HTTP` e `HTTP_PROXY` . Warning L'abilitazione `insecureS kipVerification` non è consiglia ta, in particolare per le integrazi oni con endpoint pubblici. HTTPS Se

Nome proprietà	Tipo	Descrizione
		abilitiinsecureSkipVerification , aumenti il rischio di man-in-the-middle attacchi.
serverNameToVerify	string	Supportato solo per le integrazioni HTTP API private. Se si specifica un nome di server, API Gateway lo utilizza per verificare il nome host sul certificato dell'integrazione. Il nome del server è incluso anche nell'TLShandshake per supportare Server Name Indication (SNI) o l'hosting virtuale.

x-amazon-apigateway-integration. tlsConfig esempi

Il seguente esempio di Open API 3.0 abilita `insecureSkipVerification` l'integrazione REST API HTTP del proxy.

```
"x-amazon-apigateway-integration": {
  "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
  "responses": {
     default": {
       "statusCode": "200"
      }
  },
  "passthroughBehavior": "when_no_match",
  "httpMethod": "ANY",
  "tlsConfig" : {
    "insecureSkipVerification" : true
  }
  "type": "http_proxy",
```

```
}
```

Il seguente esempio di Open API 3.0 specifica una `serverNameToVerify` per un'integrazione HTTP API privata.

```
"x-amazon-apigateway-integration" : {
  "payloadFormatVersion" : "1.0",
  "connectionId" : "abc123",
  "type" : "http_proxy",
  "httpMethod" : "ANY",
  "uri" : "arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/app/my-load-
balancer/50dc6c495c0c9188/0467ef3c8400ae65",
  "connectionType" : "VPC_LINK",
  "tlsConfig" : {
     "serverNameToVerify" : "example.com"
  }
}
```

x-amazon-apigateway-minimum-dimensione di compressione

Speciifica la dimensione minima di compressione per a. REST API Per abilitare la compressione, specificare un numero intero compreso tra 0 e 10485760. Per ulteriori informazioni, consulta Compressione del payload per REST APIs in API Gateway .

x-amazon-apigateway-minimum-compression-size, esempio

L'esempio seguente specifica una dimensione di compressione minima di byte per a. `5242880` REST API

```
"x-amazon-apigateway-minimum-compression-size": 5242880
```

x-amazon-apigateway-policy

Specifica una politica delle risorse per un. REST API Per ulteriori informazioni sulle policy delle risorse, consulta Controlla l'accesso a una politica delle risorse di REST API API With Gateway. Per esempi di policy delle risorse, consulta APIEsempi di policy sulle risorse di Gateway.

`x-amazon-apigateway-policy`Esempio

L'esempio seguente specifica una politica delle risorse per a. REST API La politica delle risorse nega (blocca) il traffico in entrata API da e verso un blocco di indirizzi IP di origine specificato. Al momento dell'importazione, `"execute-api:/*"` viene convertito in`arn:aws:execute-api:`*`region`*`:`*`account-id`*`:`*`api-id`*`/*`, utilizzando la regione corrente, l'ID AWS dell'account e l'ID corrente RESTAPI.

```
"x-amazon-apigateway-policy": {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
               "execute-api:/*"
            ],
            "Condition" : {
                "IpAddress": {
                    "aws:SourceIp": "192.0.2.0/24"
                }
            }
        }
    ]
}
```

x-amazon-apigateway-requestproprietà -validator

Speciifica un validatore di richiesta, facendo riferimento a una parte *`request_validator_name`* della x-amazon-apigateway-requestoggetto -validators mappa, per abilitare la convalida della richiesta sul contenitore o su un metodo. API Il valore di questa estensione è una stringa. JSON

Questa estensione può essere specificata a API livello o a livello di metodo. Il validatore a API livello di metodo si applica a tutti i metodi a meno che non venga sovrascritto dal validatore a livello di metodo.

`x-amazon-apigateway-request-validator` Esempio

L'esempio seguente applica il validatore della `basic` richiesta al API livello mentre applica il validatore della richiesta alla richiesta. `parameter-only` `POST /validation`

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "x-amazon-apigateway-request-validators" : {
    "basic" : {
      "validateRequestBody" : true,
      "validateRequestParameters" : true
    },
    "params-only" : {
      "validateRequestBody" : false,
      "validateRequestParameters" : true
    }
  },
  "x-amazon-apigateway-request-validator" : "basic",
  "paths": {
    "/validation": {
      "post": {
        "x-amazon-apigateway-request-validator" : "params-only",
       ...
     }
}
```

x-amazon-apigateway-requestoggetto -validators

Definisce i validatori di richiesta supportati per il contenitore API come mappa tra il nome di un validatore e le regole di convalida della richiesta associate. Questa estensione si applica a. REST API

Nome proprietà	Tipo	Descrizione
request_validator_name	x-amazon-apigateway-request-validatori. requestValidator oggetto	Specifica le regole di convalida costituite dal validatore denominato. Ad esempio: `"basic" : { "validateRequestBody" : true, "validateRequestParameters" : true },` Per applicare il validatore a un metodo specifico, fai riferimento al rispettivo nome (`basic`) come al valore della proprietà x-amazon-apigateway-request proprietà -validator.

`x-amazon-apigateway-request-validators`Esempio

L'esempio seguente mostra un set di validatori di richiesta per una mappa API as a tra il nome di un validatore e le regole di convalida della richiesta associate.

OpenAPI 2.0

```
{
  "swagger": "2.0",
  ...
  "x-amazon-apigateway-request-validators" : {
    "basic" : {
      "validateRequestBody" : true,
      "validateRequestParameters" : true
    },
    "params-only" : {
      "validateRequestBody" : false,
```

```
      "validateRequestParameters" : true
    }
  },
  ...
}
```

x-amazon-apigateway-request-validatori. requestValidator oggetto

Specifica le regole di convalida di un validatore di richieste nell'ambito della definizione della mappa x-amazon-apigateway-requestoggetto -validators.

Nome proprietà	Tipo	Descrizione
`validateRequestBody`	`Boolean`	Specifica se convalidare (`true`) o non convalidare (`false`) il corpo della richiesta .
`validateRequestParameters`	`Boolean`	Specifica se convalidare (`true`) o non convalidare (`false`) i parametri di richiesta necessari.

`x-amazon-apigateway-request-validators.requestValidator`Esempio

L'esempio seguente mostra un validatore di risposte di soli parametri.

```
"params-only": {
    "validateRequestBody" : false,
    "validateRequestParameters" : true
}
```

x-amazon-apigateway-tagproprietà -value

Speciifica il valore di un AWS tag per un. HTTP API È possibile utilizzare la `x-amazon-apigateway-tag-value` proprietà come parte dell'oggetto Open API tag a livello principale per specificare AWS i tag per un. HTTP API Se specificate il nome di un tag senza la `x-amazon-apigateway-tag-value` proprietà, API Gateway crea un tag con una stringa vuota per un valore.

Per ulteriori informazioni sul tagging, consulta Tagging delle risorse API Gateway.

Nome proprietà	Tipo	Descrizione
`name`	`String`	Specifica la chiave di tag.
`x-amazon-apigateway-tag-value`	`String`	Specifica il valore del tag.

`x-amazon-apigateway-tag-value`Esempio

L'esempio seguente specifica due tag per un HTTPAPI:

- "Owner": "Admin"
- "Prod": ""

```
"tags": [
    {
      "name": "Owner",
      "x-amazon-apigateway-tag-value": "Admin"
    },
    {
      "name": "Prod"
    }
]
```

Sicurezza in Amazon API Gateway

La sicurezza del cloud AWS è la massima priorità. In qualità di AWS cliente, puoi beneficiare di un data center e di un'architettura di rete progettati per soddisfare i requisiti delle organizzazioni più sensibili alla sicurezza.

La sicurezza è una responsabilità condivisa tra AWS te e te. Il modello di responsabilità condivisa descrive questo modello come sicurezza del cloud e sicurezza nel cloud:

- Sicurezza del cloud: AWS è responsabile della protezione dell'infrastruttura che gestisce AWS i servizi nel AWS cloud. AWS ti fornisce anche servizi che puoi utilizzare in modo sicuro. I revisori esterni testano e verificano regolarmente l'efficacia della nostra sicurezza nell'ambito dei AWS Programmi di AWS conformità dei Programmi di conformità dei di . Per informazioni sui programmi di conformità che si applicano ad Amazon API Gateway, consulta AWS i servizi nell'ambito del programma di conformità, i AWS servizi nell'ambito del programma .
- Sicurezza nel cloud: la tua responsabilità è determinata dal AWS servizio che utilizzi. Sei anche responsabile di altri fattori, tra cui la riservatezza dei dati, i requisiti della tua azienda e le leggi e normative vigenti.

Questa documentazione aiuta a comprendere come applicare il modello di responsabilità condivisa quando si usa API Gateway. Gli argomenti seguenti illustrano come configurare API Gateway per soddisfare gli obiettivi di sicurezza e conformità. Scopri anche come utilizzare altri AWS servizi che ti aiutano a monitorare e proteggere le tue risorse API Gateway.

Per ulteriori informazioni, consulta Panoramica della sicurezza in Amazon API Gateway.

Argomenti

- Protezione dei dati in Amazon API Gateway
- Gestione delle identità e degli accessi per Amazon API Gateway
- Logging e monitoraggio in Amazon API Gateway
- Convalida della conformità per Amazon API Gateway
- Resilienza in Amazon API Gateway
- Sicurezza dell'infrastruttura in Amazon API Gateway
- Analisi delle vulnerabilità in Amazon API Gateway

- Best practice di sicurezza in Amazon API Gateway

Protezione dei dati in Amazon API Gateway

Il modello di responsabilità AWS condivisa modello si applica alla protezione dei dati in Amazon API Gateway. Come descritto in questo modello, AWS è responsabile della protezione dell'infrastruttura globale che gestisce tutti i Cloud AWS. L'utente è responsabile del controllo dei contenuti ospitati su questa infrastruttura. L'utente è inoltre responsabile della configurazione della protezione e delle attività di gestione per i Servizi AWS utilizzati. Per ulteriori informazioni sulla privacy dei dati, consulta la sezione Privacy dei dati FAQ. Per informazioni sulla protezione dei dati in Europa, consulta il Modello di responsabilitàAWS condivisa e GDPR il post sul blog sulla AWS sicurezza.

Ai fini della protezione dei dati, ti consigliamo di proteggere Account AWS le credenziali e di configurare i 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.
- UsaSSL/TLSper comunicare con AWS le risorse. Richiediamo TLS 1.2 e consigliamo TLS 1.3.
- Configurazione API e registrazione delle attività degli utenti con AWS CloudTrail. Per informazioni sull'uso dei CloudTrail percorsi per registrare AWS le attività, consulta Lavorare con i CloudTrail percorsi nella Guida per l'AWS CloudTrail utente.
- Utilizza soluzioni di AWS crittografia, insieme a tutti i controlli di sicurezza predefiniti all'interno 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 hai bisogno di FIPS 140-3 moduli crittografici convalidati per accedere AWS tramite un'interfaccia a riga di comando o unAPI, usa un endpoint. FIPS Per ulteriori informazioni sugli FIPS endpoint disponibili, vedere Federal Information Processing Standard () 140-3. FIPS

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. Ciò include quando si lavora con API Gateway o altri utenti Servizi AWS utilizzando la console, API AWS CLI, o. AWS SDKs 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. Se fornisci un messaggio URL a

un server esterno, ti consigliamo vivamente di non includere le informazioni sulle credenziali URL per convalidare la tua richiesta a quel server.

Crittografia dei dati in Amazon API Gateway

Lo scopo della protezione dati è proteggere i dati sia in transito (durante la trasmissione verso e da API Gateway), sia quando sono inattivi (ovvero quando sono archiviati in AWS).

Crittografia dei dati inattivi in Amazon API Gateway

Se si sceglie di abilitare la memorizzazione nella cache per un'API REST, è possibile abilitare la crittografia della cache. Per ulteriori informazioni, consulta Impostazioni della cache per REST APIs in API Gateway.

Per ulteriori informazioni sulla protezione dei dati, consulta il post del blog AWS Modello di responsabilità condivisa e GDPR su AWS Security Blog.

Crittografia dei dati in transito in Amazon API Gateway

Le API create con Amazon API Gateway espongono solo endpoint HTTPS. API Gateway non supporta gli endpoint non crittografati (HTTP).

API Gateway gestisce i certificati per gli endpoint predefiniti `execute-api`. Se si configura un nome di dominio personalizzato, si specifica il certificato per il nome di dominio. Come best practice, non bloccare i certificati.

Per una maggiore sicurezza, è possibile scegliere una versione del protocollo Transport Layer Security (TLS) minima da applicare per il dominio API Gateway personalizzato. Le API WebSocket e le API HTTP supportano solo TLS 1.2. Per ulteriori informazioni, consulta Scegli una politica di sicurezza per il tuo dominio REST API personalizzato in API Gateway.

È inoltre possibile impostare una distribuzione Amazon CloudFront con un certificato SSL personalizzato nel proprio account e utilizzarla con API regionali. Puoi quindi configurare la policy di sicurezza per la distribuzione CloudFront con TLS 1.1 o versione successiva in base ai tuoi requisiti di sicurezza e conformità.

Per ulteriori informazioni sulla protezione dei dati, consulta Proteggi le tue API REST in API Gateway e il post del blog relativo al modello di responsabilità condivisa e GDPR AWS nel Blog sulla sicurezza AWS.

Riservatezza del traffico Internet

Utilizzando Amazon API Gateway, puoi creare API REST private cui è possibile accedere solo da Amazon Virtual Private Cloud (VPC). Il VPC utilizza un endpoint VPC di interfaccia, che è un'interfaccia di rete endpoint creata nel VPC. Utilizzando le policy delle risorse, è possibile consentire o negare l'accesso alle tue API da VPC ed endpoint VPC selezionati, inclusi gli account AWS. Ogni endpoint può essere utilizzato per l'accesso a più API private. È inoltre possibile utilizzare AWS Direct Connect per stabilire una connessione da una rete on-premise ad Amazon VPC e accedere all'API privata durante tale connessione. In tutti i casi, il traffico alla tua API privata utilizza connessioni sicure e non lascia la rete Amazon; è isolato dall'Internet pubblico. Per ulteriori informazioni, consulta the section called “Privato REST APIs”.

Gestione delle identità e degli accessi per Amazon API Gateway

AWS Identity and Access Management (IAM) è un dispositivo Servizio AWS che aiuta un amministratore a controllare in modo sicuro l'accesso alle AWS risorse. IAMgli amministratori controllano chi può essere autenticato (effettuato l'accesso) e autorizzato (disporre delle autorizzazioni) a utilizzare le risorse Gateway. API IAMè un software Servizio AWS che puoi utilizzare senza costi aggiuntivi.

Argomenti

- Destinatari
- Autenticazione con identità
- Gestione dell'accesso con policy
- Come funziona Amazon API Gateway con IAM
- Esempi di policy API basate sull'identità di Amazon Gateway
- Esempi di policy API basate sulle risorse di Amazon Gateway
- Risoluzione dei problemi relativi all'identità e all'accesso ad Amazon API Gateway
- Utilizzo dei ruoli collegati ai servizi per API Gateway

Destinatari

Il modo in cui usi AWS Identity and Access Management (IAM) varia a seconda del lavoro svolto in API Gateway.

Utente del servizio: se utilizzi il servizio API Gateway per svolgere il tuo lavoro, l'amministratore ti fornisce le credenziali e le autorizzazioni necessarie. Man mano che utilizzi più funzionalità di API Gateway per svolgere il tuo lavoro, potresti aver bisogno di autorizzazioni aggiuntive. La comprensione della gestione dell'accesso ti consente di richiedere le autorizzazioni corrette all'amministratore. Se non riesci ad accedere a una funzionalità di API Gateway, consultaRisoluzione dei problemi relativi all'identità e all'accesso ad Amazon API Gateway.

Amministratore del servizio: se sei responsabile delle risorse API Gateway della tua azienda, probabilmente hai pieno accesso a API Gateway. È tuo compito determinare a quali funzionalità e risorse del API Gateway gli utenti del servizio devono accedere. È quindi necessario inviare richieste all'IAMamministratore per modificare le autorizzazioni degli utenti del servizio. Consulta le informazioni contenute in questa pagina per comprendere i concetti di base diIAM. Per ulteriori informazioni su come la tua azienda può utilizzare IAM API Gateway, consultaCome funziona Amazon API Gateway con IAM.

IAMamministratore: se sei un IAM amministratore, potresti voler conoscere i dettagli su come scrivere politiche per gestire l'accesso a API Gateway. Per visualizzare esempi di policy basate sull'identità di API Gateway che puoi utilizzare inIAM, consulta. Esempi di policy API basate sull'identità di Amazon Gateway

Autenticazione con identità

L'autenticazione è il modo in cui accedi AWS utilizzando le tue credenziali di identità. È necessario autenticarsi (accedere a AWS) come Utente root dell'account AWS, come IAM utente o assumendo un ruolo. IAM

È possibile accedere AWS come identità federata utilizzando le credenziali fornite tramite una fonte di identità. AWS IAM Identity Center Gli utenti (IAMIdentity Center), l'autenticazione Single Sign-On della tua azienda e le tue credenziali di Google o Facebook sono esempi di identità federate. Quando accedi come identità federata, l'amministratore aveva precedentemente configurato la federazione delle identità utilizzando i ruoli. IAM Quando si accede AWS utilizzando la federazione, si assume indirettamente un ruolo.

A seconda del tipo di utente, puoi accedere al AWS Management Console o al portale di AWS accesso. Per ulteriori informazioni sull'accesso a AWS, vedi Come accedere al tuo Account AWS nella Guida per l'Accedi ad AWS utente.

Se accedi a AWS livello di codice, AWS fornisce un kit di sviluppo software (SDK) e un'interfaccia a riga di comando () per firmare crittograficamente le tue richieste utilizzando le tue credenziali. CLI Se

non utilizzi AWS strumenti, devi firmare tu stesso le richieste. Per ulteriori informazioni sull'utilizzo del metodo consigliato per firmare autonomamente le richieste, consulta Firmare AWS API le richieste nella Guida per l'IAMutente.

A prescindere dal metodo di autenticazione utilizzato, potrebbe essere necessario specificare ulteriori informazioni sulla sicurezza. Ad esempio, ti AWS consiglia di utilizzare l'autenticazione a più fattori (MFA) per aumentare la sicurezza del tuo account. Per ulteriori informazioni, consulta Autenticazione a più fattori nella Guida per l'AWS IAM Identity Center utente e Utilizzo dell'autenticazione a più fattori (MFA) AWS nella Guida per l'IAMutente.

Account AWS utente root

Quando si crea un account Account AWS, si inizia con un'identità di accesso che ha accesso completo a tutte Servizi AWS le risorse dell'account. Questa identità è denominata utente Account AWS root ed è accessibile effettuando l'accesso 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 utilizzale per eseguire le operazioni che solo l'utente root può eseguire. Per l'elenco completo delle attività che richiedono l'accesso come utente root, consulta Attività che richiedono le credenziali dell'utente root nella Guida per l'IAMutente.

IAM users and groups

Un IAMutente è un'identità interna all'utente Account AWS che dispone di autorizzazioni specifiche per una singola persona o applicazione. Laddove possibile, consigliamo di fare affidamento su credenziali temporanee anziché creare IAM utenti con credenziali a lungo termine come password e chiavi di accesso. Tuttavia, se hai casi d'uso specifici che richiedono credenziali a lungo termine con IAM gli utenti, ti consigliamo di ruotare le chiavi di accesso. Per ulteriori informazioni, consulta Ruotare regolarmente le chiavi di accesso per i casi d'uso che richiedono credenziali a lungo termine nella Guida per l'utente. IAM

Un IAMgruppo 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, potresti avere un gruppo denominato IAMAdminse concedere a quel gruppo le autorizzazioni per IAM amministrare le risorse.

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 IAM utente (anziché un ruolo) nella Guida per l'IAMutente.

IAMruoli

Un IAMruolo è un'identità interna all'utente Account AWS che dispone di autorizzazioni specifiche. È simile a un IAM utente, ma non è associato a una persona specifica. È possibile assumere temporaneamente un IAM ruolo in AWS Management Console cambiando ruolo. È possibile assumere un ruolo chiamando un' AWS APIoperazione AWS CLI or o utilizzando un'operazione personalizzataURL. Per ulteriori informazioni sui metodi di utilizzo dei ruoli, vedere Utilizzo IAM dei ruoli nella Guida per l'IAMutente.

IAMi ruoli 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 informazioni sui ruoli per la federazione, vedere Creazione di un ruolo per un provider di identità di terze parti nella Guida per l'IAMutente. Se utilizzi IAM Identity Center, configuri un set di autorizzazioni. Per controllare a cosa possono accedere le identità dopo l'autenticazione, IAM Identity Center correla il set di autorizzazioni a un ruolo in. IAM Per informazioni sui set di autorizzazioni, consulta Set di autorizzazioni nella Guida per l'utente di AWS IAM Identity Center .
- Autorizzazioni IAM utente temporanee: un IAM utente o un ruolo può assumere il IAM ruolo di assumere temporaneamente autorizzazioni diverse per un'attività specifica.
- Accesso su più account: puoi utilizzare un IAM ruolo per consentire a qualcuno (un responsabile fidato) di un altro account di accedere alle risorse del tuo account. I ruoli sono lo strumento principale per concedere l'accesso multi-account. Tuttavia, con alcuni Servizi AWS, è possibile allegare una policy direttamente a una risorsa (anziché utilizzare un ruolo come proxy). Per conoscere la differenza tra ruoli e politiche basate sulle risorse per l'accesso tra account diversi, consulta la sezione Accesso alle risorse su più account IAM nella Guida per l'utente. IAM
- Accesso tra servizi: alcuni Servizi AWS utilizzano funzionalità in altri. Servizi AWS Ad esempio, quando effettui una chiamata in un servizio, è normale che quel 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.
  - Sessioni di accesso inoltrato (FAS): quando utilizzi un IAM utente o un ruolo per eseguire azioni AWS, sei considerato un principale. Quando si utilizzano alcuni servizi, è possibile eseguire

un'operazione che attiva un'altra operazione in un servizio diverso. FASutilizza le autorizzazioni del principale che chiama an Servizio AWS, in combinazione con la richiesta di effettuare richieste Servizio AWS ai servizi downstream. FASle richieste vengono effettuate solo quando un servizio riceve una richiesta che richiede interazioni con altri Servizi AWS o risorse per essere completata. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le azioni. Per i dettagli FAS delle politiche relative alle richieste, consulta Forward access sessions.

- Ruolo di servizio: un ruolo di servizio è un IAMruolo che un servizio assume per eseguire azioni per conto dell'utente. Un IAM amministratore può creare, modificare ed eliminare un ruolo di servizio dall'internoIAM. Per ulteriori informazioni, vedere Creazione di un ruolo per delegare le autorizzazioni a un utente Servizio AWS nella Guida per l'IAMutente.
- 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 al servizio vengono visualizzati nel tuo account Account AWS e sono di proprietà del servizio. Un IAM amministratore può visualizzare, ma non modificare le autorizzazioni per i ruoli collegati al servizio.

- Applicazioni in esecuzione su Amazon EC2: puoi utilizzare un IAM ruolo per gestire le credenziali temporanee per le applicazioni in esecuzione su un'EC2istanza e che effettuano AWS CLI o richiedono AWS API. Ciò è preferibile alla memorizzazione delle chiavi di accesso all'interno dell'EC2istanza. Per assegnare un AWS ruolo a un'EC2istanza e renderlo disponibile per tutte le sue applicazioni, create un profilo di istanza collegato all'istanza. Un profilo di istanza contiene il ruolo e consente ai programmi in esecuzione sull'EC2istanza di ottenere credenziali temporanee. Per ulteriori informazioni, consulta Usare un IAM ruolo per concedere le autorizzazioni alle applicazioni in esecuzione su EC2 istanze Amazon nella Guida per l'IAMutente.

Per sapere se utilizzare IAM ruoli o IAM utenti, consulta Quando creare un IAM ruolo (anziché un utente) nella Guida per l'IAMutente.

Gestione dell'accesso con policy

Puoi controllare l'accesso AWS creando policy e associandole a AWS identità o risorse. Una policy è un oggetto AWS che, se associato a un'identità o a una risorsa, ne definisce le autorizzazioni. AWS valuta queste politiche quando un principale (utente, utente root o sessione di ruolo) effettua una richiesta. Le autorizzazioni nelle policy determinano l'approvazione o il rifiuto della richiesta. La maggior parte delle politiche viene archiviata AWS come JSON documenti. Per ulteriori informazioni sulla struttura e il contenuto dei documenti relativi alle JSON politiche, vedere Panoramica delle JSON politiche nella Guida per l'IAMutente.

Gli amministratori possono utilizzare AWS JSON le politiche per specificare chi ha accesso a cosa. 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 IAM amministratore può creare IAM politiche. L'amministratore può quindi aggiungere le IAM politiche ai ruoli e gli utenti possono assumerli.

IAMle politiche definiscono le autorizzazioni per un'azione indipendentemente dal metodo utilizzato per eseguire l'operazione. Ad esempio, supponiamo di disporre di una policy che consente l'operazione `iam:GetRole`. Un utente con tale criterio può ottenere informazioni sul ruolo da AWS Management Console, da o da. AWS CLI AWS API

Policy basate su identità

I criteri basati sull'identità sono documenti relativi alle politiche di JSON autorizzazione che è possibile allegare a un'identità, ad esempio un IAM utente, un gruppo di utenti o un ruolo. Tali policy definiscono le azioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una politica basata sull'identità, consulta Creazione di politiche nella Guida per l'utente. IAM IAM

Le policy basate su identità possono essere ulteriormente classificate come policy inline o policy gestite. Le policy inline sono integrate direttamente in un singolo utente, gruppo o ruolo. Le politiche gestite sono politiche autonome che puoi allegare a più utenti, gruppi e ruoli all'interno del tuo. Account AWS Le politiche gestite includono politiche AWS gestite e politiche gestite dai clienti. Per informazioni su come scegliere tra una politica gestita o una politica in linea, consulta Scelta tra politiche gestite e politiche in linea nella Guida per l'IAMutente.

Policy basate su risorse

Le politiche basate sulle risorse sono documenti di JSON policy allegati a una risorsa. Esempi di politiche basate sulle risorse sono le policy di trust dei IAM ruoli e le policy dei bucket di Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. Quando è collegata 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 politiche AWS gestite IAM in una politica basata sulle risorse.

Elenchi di controllo degli accessi () ACLs

Le liste di controllo degli accessi (ACLs) controllano quali principali (membri dell'account, utenti o ruoli) dispongono delle autorizzazioni per accedere a una risorsa. ACLssono simili alle politiche basate sulle risorse, sebbene non utilizzino il formato del documento di policy. JSON

Amazon S3 e Amazon VPC sono esempi di servizi che supportano. AWS WAF ACLs Per ulteriori informazioniACLs, consulta la panoramica di Access control list (ACL) nella Amazon Simple Storage Service Developer Guide.

Altri tipi di policy

AWS supporta tipi di policy aggiuntivi e 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 di autorizzazioni è una funzionalità avanzata in cui si impostano le autorizzazioni massime che una politica basata sull'identità può concedere a un'entità (utente o ruolo). IAM 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 nella Guida per l'utente. IAM IAM
- Politiche di controllo del servizio (SCPs): SCPs sono JSON politiche che specificano le autorizzazioni massime per un'organizzazione o un'unità organizzativa (OU) in. AWS Organizations AWS Organizations è un servizio per il raggruppamento e la gestione centralizzata di più Account AWS di proprietà dell'azienda. Se abiliti tutte le funzionalità di un'organizzazione, puoi applicare le politiche di controllo del servizio (SCPs) a uno o tutti i tuoi account. SCPLimita le autorizzazioni per le entità negli account dei membri, inclusa ciascuna Utente root dell'account AWS. Per ulteriori informazioni su Organizations andSCPs, consulta le politiche di controllo dei servizi nella Guida AWS Organizations per l'utente.
- 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 le politiche di sessione nella Guida IAM per l'utente.

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 determinare se consentire una richiesta quando sono coinvolti più tipi di policy, consulta Logica di valutazione delle politiche nella Guida per l'IAMutente.

Come funziona Amazon API Gateway con IAM

Prima di utilizzare IAM per gestire l'accesso a API Gateway, è necessario comprendere quali IAM funzionalità sono disponibili per l'uso con API Gateway. Per una panoramica generale del funzionamento di API Gateway e di altri AWS serviziIAM, consulta AWS Services That Work with IAM nella Guida per l'IAMutente.

Argomenti

- APIPolitiche basate sull'identità di Gateway
- APIPolitiche basate sulle risorse di Gateway
- Autorizzazione basata sui tag Gateway API
- APIRuoli Gateway IAM

APIPolitiche basate sull'identità di Gateway

Con le politiche IAM basate sull'identità, è possibile specificare quali azioni e risorse sono consentite o negate, nonché le condizioni in base alle quali le azioni sono consentite o negate. APIGateway supporta azioni, risorse e chiavi di condizione specifiche. Per ulteriori informazioni sulle azioni, le risorse e le chiavi di condizione API specifiche del gateway, consulta Azioni, risorse e chiavi di condizione per Amazon API Gateway Management e Azioni, risorse e chiavi di condizione per Amazon API Gateway Management V2. Per informazioni su tutti gli elementi utilizzati in una JSON policy, consulta Policy Elements Reference nella IAM JSON Guida per l'utente. IAM

L'esempio seguente mostra una politica basata sull'identità che consente a un utente di creare o aggiornare solo informazioni private. REST APIs Per ulteriori esempi, consulta the section called “Esempi di policy basate su identità”.

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Sid": "ScopeToPrivateApis",
      "Effect": "Allow",
      "Action": [
        "apigateway:PATCH",
        "apigateway:POST",
        "apigateway:PUT"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/restapis",
        "arn:aws:apigateway:us-east-1::/restapis/??????????"
      ],
      "Condition": {
        "ForAllValues:StringEqualsIfExists": {
          "apigateway:Request/EndpointType": "PRIVATE",
          "apigateway:Resource/EndpointType": "PRIVATE"
        }
      }
    },
    {
      "Sid": "AllowResourcePolicyUpdates",
      "Effect": "Allow",
      "Action": [
          "apigateway:UpdateRestApiPolicy"
      ],
      "Resource": [
          "arn:aws:apigateway:us-east-1::/restapis/*"
      ]
    }
  ]
}
```

Azioni

L'`Action`elemento di una JSON policy descrive le azioni che è possibile utilizzare per consentire o negare l'accesso a una policy.

Le azioni politiche in API Gateway utilizzano il seguente prefisso prima dell'azione:. `apigateway:` Le istruzioni della policy devono includere un elemento `Action` o `NotAction`. APIGateway definisce il proprio set di azioni che descrivono le attività che è possibile eseguire con questo servizio.

L'`Action`espressione API -managing ha il formato`apigateway:`*`action`*, dove *`action`* è una delle seguenti azioni API Gateway: GETPOST, PUT, DELETE,, PATCH(per aggiornare le risorse) o*, che rappresenta tutte le azioni precedenti.

Alcuni esempi dell'espressione `Action` includono:

- **`apigateway:*`**per tutte le azioni API Gateway.
- **`apigateway:GET`**solo per l'GETazione in API Gateway.

Per specificare più azioni in una sola istruzione, separa ciascuna di esse con una virgola come mostrato di seguito:

```
"Action": [
      "apigateway:action1",
      "apigateway:action2"
```

Per informazioni sui HTTP verbi da utilizzare per operazioni API gateway specifiche, consulta Amazon API Gateway Version 1 API Reference (RESTAPIs) e Amazon API Gateway Version 2 API Reference (WebSocket and HTTPAPIs).

Per ulteriori informazioni, consulta the section called “Esempi di policy basate su identità”.

Risorse

Gli amministratori possono utilizzare AWS JSON le policy per specificare chi ha accesso a cosa. Cioè, quale principale può eseguire operazioni su quali risorse, e in quali condizioni.

L'elemento `Resource` JSON policy specifica l'oggetto o gli oggetti a cui si applica l'azione. Le istruzioni devono includere un elemento `Resource`o un elemento `NotResource`. Come best practice, specifica una risorsa utilizzando il relativo Amazon Resource Name (ARN). Puoi eseguire questa operazione per azioni che supportano un tipo di risorsa specifico, note come autorizzazioni a livello di risorsa.

Per le azioni 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": "*"
```

APILe risorse del gateway hanno il seguente ARN formato:

```
arn:aws:apigateway:region::resource-path-specifier
```

Ad esempio, per specificare a REST API con l'id *api-id* e le relative risorse secondarie, come gli autorizzatori presenti nella dichiarazione, utilizzano quanto segue: ARN

```
"Resource": "arn:aws:apigateway:us-east-2::/restapis/api-id/*"
```

Per specificare tutte le risorse REST APIs e le sottorisorse che appartengono a un account specifico, usa il carattere jolly (*):

```
"Resource": "arn:aws:apigateway:us-east-2::/restapis/*"
```

Per un elenco dei tipi di risorse API Gateway e relativiARNs, vedere. Documentazione di riferimento Amazon Resource Name (ARN) API Gateway

Chiavi di condizione

Gli amministratori possono utilizzare AWS JSON le policy 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 in vigore. L'elemento `Condition`nè 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 si specificano più valori per una singola chiave di condizione, AWS valuta la condizione utilizzando un'operazione logica`OR`. 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, è possibile concedere a un IAM utente l'autorizzazione ad accedere a una risorsa solo se è contrassegnata con il suo nome IAM utente. Per ulteriori informazioni, consulta gli elementi IAM della politica: variabili e tag nella Guida IAM per l'utente.

AWS supporta chiavi di condizione globali e chiavi di condizione specifiche del servizio. Per visualizzare tutte le chiavi di condizione AWS globali, consulta le chiavi di contesto delle condizioni AWS globali nella Guida per l'IAMutente.

APIGateway definisce il proprio set di chiavi di condizione e supporta anche l'utilizzo di alcune chiavi di condizione globali. Per un elenco delle chiavi di condizione di API Gateway, consulta Condition Keys for Amazon API Gateway nella Guida per l'IAMutente. Per informazioni su quali azioni e risorse puoi utilizzare con una chiave di condizione, consulta Actions Defined by Amazon API Gateway.

Per informazioni sull'assegnazione di tag, incluso il controllo degli accessi basato sugli attributi, consulta Applicazione di tag.

Esempi

Per esempi di politiche basate sull'identità di API Gateway, consulta. Esempi di policy API basate sull'identità di Amazon Gateway

APIPolitiche basate sulle risorse di Gateway

Le politiche basate sulle risorse sono documenti di JSON policy che specificano quali azioni uno specifico principale può eseguire sulla risorsa API Gateway e in quali condizioni. APIGateway supporta politiche di autorizzazione basate sulle risorse per. REST APIs Le politiche delle risorse vengono utilizzate per controllare chi può richiamare un. REST API Per ulteriori informazioni, consulta the section called “Utilizza le politiche delle risorse del API Gateway”.

Esempi

Per esempi di politiche basate sulle risorse di API Gateway, vedere. APIEsempi di policy sulle risorse di Gateway

Autorizzazione basata sui tag Gateway API

È possibile allegare tag alle risorse API Gateway o passare i tag in una richiesta a API Gateway. Per controllare l'accesso basato su tag, fornisci informazioni sui tag nell'elemento condizione di una policy utilizzando le chiavi di condizione `apigateway:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* o `aws:TagKeys`. Per ulteriori informazioni sull'etichettatura delle risorse API Gateway, consultathe section called “Controllo dell'accesso basato sugli attributi”.

Per esempi di policy basate su identità per limitare l'accesso a una risorsa in base ai tag di tale risorsa, consulta Utilizzo dei tag per controllare l'accesso alle risorse REST API di Gateway API.

APIRuoli Gateway IAM

Un IAMruolo è un'entità all'interno del tuo AWS account che dispone di autorizzazioni specifiche.

Utilizzo di credenziali temporanee con Gateway API

È possibile utilizzare credenziali temporanee per accedere con la federazione, assumere un IAM ruolo o assumere un ruolo tra account. È possibile ottenere credenziali di sicurezza temporanee chiamando AWS STS API operazioni come o. AssumeRoleGetFederationToken

APIGateway supporta l'utilizzo di credenziali temporanee.

Ruoli collegati ai servizi

I ruoli collegati ai AWS servizi consentono ai servizi di accedere alle risorse di altri servizi per completare un'azione per conto dell'utente. I ruoli collegati ai servizi vengono visualizzati nell'IAMaccount e sono di proprietà del servizio. Un IAM amministratore può visualizzare ma non modificare le autorizzazioni per i ruoli collegati al servizio.

APIGateway supporta ruoli collegati ai servizi. Per informazioni sulla creazione o la gestione di ruoli collegati ai servizi API Gateway, vedere. Utilizzo dei ruoli collegati ai servizi per API Gateway

Ruoli di servizio

Un servizio può assumere un ruolo del servizio a tuo nome. Un ruolo del servizio consente al servizio di accedere alle risorse in altri servizi per completare un'operazione per conto dell'utente. I ruoli di servizio vengono visualizzati nell'IAMaccount e sono di proprietà dell'account, quindi un amministratore può modificare le autorizzazioni per questo ruolo. Tuttavia, il farlo potrebbe pregiudicare la funzionalità del servizio.

APIGateway supporta i ruoli di servizio.

Esempi di policy API basate sull'identità di Amazon Gateway

Per impostazione predefinita, IAM gli utenti e i ruoli non sono autorizzati a creare o modificare API risorse Gateway. Inoltre, non possono eseguire attività utilizzando AWS Management Console AWS CLI, o AWS SDKs. Un IAM amministratore deve creare IAM politiche che concedano a utenti e ruoli l'autorizzazione a eseguire API operazioni specifiche sulle risorse specifiche di cui ha bisogno. L'amministratore deve quindi allegare tali politiche agli IAM utenti o ai gruppi che richiedono tali autorizzazioni.

Per informazioni su come creare IAM politiche, vedere Creazione di politiche nella JSON scheda nella Guida per l'IAMutente. Per informazioni sulle azioni, le risorse e le condizioni specifiche di API Gateway, consulta Azioni, risorse e chiavi di condizione per Amazon API Gateway Management e Azioni, risorse e chiavi di condizione per Amazon API Gateway Management V2.

Argomenti

- Best practice per le policy
- Consentire agli utenti di visualizzare le loro autorizzazioni
- Autorizzazioni di lettura semplici
- Crea solo i nostri autorizzatori REQUEST JWT
- Richiedere che l'endpoint predefinito execute-api sia disabilitato
- Consenti agli utenti di creare o aggiornare solo dati privati REST APIs
- Richiedi che i percorsi abbiano l'autorizzazione API
- Impedisci a un utente di creare o aggiornare un VPC link

Best practice per le policy

Le politiche basate sull'identità determinano se qualcuno può creare, accedere o eliminare le risorse API Gateway nel tuo account. Queste azioni possono comportare costi aggiuntivi per l' Account AWS. Quando crei o modifichi policy basate su identità, segui queste linee guida e raccomandazioni:

- Inizia con le policy AWS gestite e passa alle autorizzazioni con privilegi minimi: per iniziare a concedere autorizzazioni a utenti e carichi di lavoro, utilizza le politiche gestite che concedono le autorizzazioni per molti casi d'uso comuni.AWS Sono disponibili nel tuo. Account AWS Ti consigliamo di ridurre ulteriormente le autorizzazioni definendo politiche gestite dai AWS clienti specifiche per i tuoi casi d'uso. Per ulteriori informazioni, consulta le politiche AWS gestite o le politiche AWS gestite per le funzioni lavorative nella Guida per l'IAMutente.
- Applica le autorizzazioni con privilegi minimi: quando imposti le autorizzazioni con le IAM politiche, concedi solo le autorizzazioni necessarie 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 per applicare le autorizzazioni, consulta Politiche e autorizzazioni nella Guida IAM per l'utente. IAM IAM
- Utilizza le condizioni nelle IAM politiche per limitare ulteriormente l'accesso: puoi aggiungere una condizione alle tue politiche per limitare l'accesso ad azioni e risorse. Ad esempio, puoi scrivere una condizione di policy per specificare che tutte le richieste devono essere inviate utilizzandoSSL. È inoltre possibile utilizzare condizioni per concedere l'accesso alle azioni di servizio se vengono utilizzate tramite uno specifico Servizio AWS, ad esempio AWS CloudFormation. Per ulteriori informazioni, consulta Elementi IAM JSON della politica: Condizione nella Guida IAM per l'utente.
- Usa IAM Access Analyzer per convalidare IAM le tue policy e garantire autorizzazioni sicure e funzionali: IAM Access Analyzer convalida le policy nuove ed esistenti in modo che aderiscano

al linguaggio delle IAM policy () e alle best practice. JSON IAM IAMAccess Analyzer fornisce più di 100 controlli delle politiche e consigli pratici per aiutarti a creare policy sicure e funzionali. Per ulteriori informazioni, vedere Convalida delle policy di IAM Access Analyzer nella Guida per l'utente. IAM

- Richiedi l'autenticazione a più fattori (MFA): se hai uno scenario che richiede l'utilizzo di IAM utenti o di un utente root Account AWS, attiva questa opzione MFA per una maggiore sicurezza. Per richiedere MFA quando vengono richiamate API le operazioni, aggiungi MFA delle condizioni alle tue politiche. Per ulteriori informazioni, vedere Configurazione dell'APIaccesso MFA protetto nella Guida per l'IAMutente.

Per ulteriori informazioni sulle procedure consigliate inIAM, consulta la sezione Procedure consigliate in materia di sicurezza IAM nella Guida per l'IAMutente.

Consentire agli utenti di visualizzare le loro autorizzazioni

Questo esempio mostra come è possibile creare una politica che consenta IAM agli utenti di visualizzare le politiche in linea e gestite allegate alla loro identità utente. Questa politica include le autorizzazioni per completare questa azione sulla console o utilizzando o a livello di codice. AWS CLI AWS API

```
{
    "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": "*"
        }
    ]
}
```

Autorizzazioni di lettura semplici

Questa politica di esempio fornisce all'utente il permesso di ottenere informazioni su tutte le risorse di un HTTP o WebSocket API con l'identificatore di `a123456789` nella AWS regione us-east-1. La risorsa `arn:aws:apigateway:us-east-1::/apis/a123456789/*` include tutte le risorse secondarie API come gli autorizzatori e le distribuzioni.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apigateway:GET"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis/a123456789/*"
      ]
    }
  ]
}
```

Crea solo i nostri autorizzatori REQUEST JWT

Questa politica di esempio consente a un utente di creare solo APIs con `REQUEST` o senza `JWT` autorizzazioni, anche tramite importazione. Nella `Resource` sezione della politica, `arn:aws:apigateway:us-east-1::/apis/??????????` richiede che le risorse abbiano un massimo di 10 caratteri, il che esclude le sottorisorse di un. API Questo esempio utilizza

`ForAllValues` nella `Condition` sezione perché gli utenti possono creare più autorizzatori contemporaneamente importando un. API

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "OnlyAllowSomeAuthorizerTypes",
      "Effect": "Allow",
      "Action": [
        "apigateway:PUT",
        "apigateway:POST",
        "apigateway:PATCH"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis",
        "arn:aws:apigateway:us-east-1::/apis/??????????",
        "arn:aws:apigateway:us-east-1::/apis/*/authorizers",
        "arn:aws:apigateway:us-east-1::/apis/*/authorizers/*"
      ],
      "Condition": {
        "ForAllValues:StringEqualsIfExists": {
          "apigateway:Request/AuthorizerType": [
            "REQUEST",
            "JWT"
          ]
        }
      }
    }
  ]
}
```

Richiedere che l'endpoint predefinito **`execute-api`** sia disabilitato

Questo criterio di esempio consente agli utenti di creare, aggiornare o importare un fileAPI, con il seguente requisito`DisableExecuteApiEndpoint`. `true` In `DisableExecuteApiEndpoint` tal caso`true`, i client non possono utilizzare l'`execute-api`endpoint predefinito per richiamare un. API

Utilizziamo la `BoolIfExists` condizione per gestire una chiamata di aggiornamento in API cui non è stata compilata la chiave di `DisableExecuteApiEndpoint` condizione. Quando un utente tenta di creare o importare unaAPI, la chiave `DisableExecuteApiEndpoint` condizionale viene sempre compilata.

Poiché la `apis/*` risorsa acquisisce anche risorse secondarie come autorizzatori o metodi, la estendiamo esplicitamente solo con una dichiarazione. APIs `Deny`

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DisableExecuteApiEndpoint",
      "Effect": "Allow",
      "Action": [
        "apigateway:PATCH",
        "apigateway:POST",
        "apigateway:PUT"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis",
        "arn:aws:apigateway:us-east-1::/apis/*"
      ],
      "Condition": {
        "BoolIfExists": {
          "apigateway:Request/DisableExecuteApiEndpoint": true
        }
      }
    },
    {
      "Sid": "ScopeDownToJustApis",
      "Effect": "Deny",
      "Action": [
        "apigateway:PATCH",
        "apigateway:POST",
        "apigateway:PUT"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis/*/*"
      ]
    }
  ]
}
```

Consenti agli utenti di creare o aggiornare solo dati privati REST APIs

Questo criterio di esempio utilizza le chiavi di condizione per richiedere che un utente crei solo `PRIVATE` APIs e per impedire aggiornamenti che potrebbero modificare un formato API da `PRIVATE` a un altro tipo, ad esempio`REGIONAL`.

Di solito `ForAllValues` richiediamo che ogni `EndpointType` aggiunta a un API sia`PRIVATE`. Utilizziamo una chiave di condizione della risorsa per consentire qualsiasi aggiornamento a un API purché lo sia`PRIVATE`. `ForAllValues`si applica solo se è presente una chiave di condizione.

Utilizziamo il non-greedy matcher (?) per confrontare in modo esplicito e impedire l'utilizzo di API risorse diverse API IDs da quelle consentite, come gli autorizzatori.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ScopePutToPrivateApis",
            "Effect": "Allow",
            "Action": [
                "apigateway:PUT"
            ],
            "Resource": [
                "arn:aws:apigateway:us-east-1::/restapis",
                "arn:aws:apigateway:us-east-1::/restapis/??????????"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "apigateway:Resource/EndpointType": "PRIVATE"
                }
            }
        },
        {
            "Sid": "ScopeToPrivateApis",
            "Effect": "Allow",
            "Action": [
                "apigateway:DELETE",
                "apigateway:PATCH",
                "apigateway:POST"
            ],
            "Resource": [
                "arn:aws:apigateway:us-east-1::/restapis",
                "arn:aws:apigateway:us-east-1::/restapis/??????????"
```

```
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "apigateway:Request/EndpointType": "PRIVATE",
                    "apigateway:Resource/EndpointType": "PRIVATE"
                }
            }
        },
        {
            "Sid": "AllowResourcePolicyUpdates",
            "Effect": "Allow",
            "Action": [
                "apigateway:UpdateRestApiPolicy"
            ],
            "Resource": [
                "arn:aws:apigateway:us-east-1::/restapis/*"
            ]
        }
    ]
}
```

Richiedi che i percorsi abbiano l'autorizzazione API

Questa policy fa sì che i tentativi di creare o aggiornare un route (anche attraverso l'importazione) non riescono se il route non dispone di autorizzazione. `ForAnyValue` restituisce false se la chiave non è presente, ad esempio quando un route non viene creato o aggiornato. Usiamo `ForAnyValue` perché più route possono essere creati attraverso l'importazione.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowUpdatesOnApisAndRoutes",
      "Effect": "Allow",
      "Action": [
        "apigateway:POST",
        "apigateway:PATCH",
        "apigateway:PUT"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis",
        "arn:aws:apigateway:us-east-1::/apis/??????????",
        "arn:aws:apigateway:us-east-1::/apis/*/routes",
```

```
        "arn:aws:apigateway:us-east-1::/apis/*/routes/*"
      ]
    },
    {
      "Sid": "DenyUnauthorizedRoutes",
      "Effect": "Deny",
      "Action": [
        "apigateway:POST",
        "apigateway:PATCH",
        "apigateway:PUT"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis",
        "arn:aws:apigateway:us-east-1::/apis/*"
      ],
      "Condition": {
        "ForAnyValue:StringEqualsIgnoreCase": {
          "apigateway:Request/RouteAuthorizationType": "NONE"
        }
      }
    }
  ]
}
```

Impedisci a un utente di creare o aggiornare un VPC link

Questa politica impedisce a un utente di creare o aggiornare un VPC link. Un VPC link ti consente di esporre le risorse all'interno di Amazon VPC a clienti esterni a. VPC

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyVPCLink",
      "Effect": "Deny",
      "Action": [
        "apigateway:POST",
        "apigateway:PUT",
        "apigateway:PATCH"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/vpclinks",
        "arn:aws:apigateway:us-east-1::/vpclinks/*"
```

```
        ]
      }
    ]
}
```

Esempi di policy API basate sulle risorse di Amazon Gateway

Per esempi di policy basate su risorse, consult the section called “APIEsempi di policy sulle risorse di Gateway”.

Risoluzione dei problemi relativi all'identità e all'accesso ad Amazon API Gateway

Utilizza le seguenti informazioni per aiutarti a diagnosticare e risolvere i problemi più comuni che potresti riscontrare quando lavori con API Gateway eIAM.

Argomenti

- Non sono autorizzato a eseguire un'azione in API Gateway
- Non sono autorizzato a eseguire iam: PassRole
- Voglio consentire a persone esterne al mio AWS account di accedere alle mie risorse API Gateway

Non sono autorizzato a eseguire un'azione in API Gateway

Se ricevi un errore che indica che non disponi dell'autorizzazione per eseguire un'operazione, le tue policy devono essere aggiornate in modo che ti sei consentito eseguire tale operazione.

L'errore di esempio seguente si verifica quando l'`mateojacksonIAM`utente tenta di utilizzare la console per visualizzare i dettagli su una *`my-example-widget`* risorsa fittizia ma non dispone delle autorizzazioni fittizie`apigateway:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 apigateway:GetWidget on resource: my-example-widget
```

In questo caso, la policy per l'utente `mateojackson` deve essere aggiornata per consentire l'accesso alla risorsa *`my-example-widget`* utilizzando l'azione `apigateway:`*`GetWidget`*.

Se hai bisogno di assistenza, contatta l'amministratore. AWS L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Non sono autorizzato a eseguire iam: PassRole

Se ricevi un messaggio di errore indicante che non sei autorizzato a eseguire l'`iam:PassRole`azione, le tue politiche devono essere aggiornate per consentirti di trasferire un ruolo a API Gateway.

Alcuni Servizi AWS consentono di trasferire un ruolo esistente a quel servizio anziché creare un nuovo ruolo di servizio o un ruolo collegato al servizio. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per trasmettere il ruolo al servizio.

L'errore di esempio seguente si verifica quando un IAM utente denominato `marymajor` tenta di utilizzare la console per eseguire un'azione in API Gateway. Tuttavia, l'azione 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`.

Se hai bisogno di assistenza, contatta AWS l'amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Voglio consentire a persone esterne al mio AWS account di accedere alle mie risorse API Gateway

È possibile creare un ruolo con il quale utenti in altri account o persone esterne all'organizzazione possono accedere alle tue risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per i servizi che supportano politiche basate sulle risorse o liste di controllo degli accessi (ACLs), puoi utilizzare tali politiche per concedere alle persone l'accesso alle tue risorse.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- Per sapere se API Gateway supporta queste funzionalità, consulta. Come funziona Amazon API Gateway con IAM
- Per informazioni su Account AWS come fornire l'accesso alle risorse di tua proprietà, consulta Fornire l'accesso a un IAM utente di un altro Account AWS utente di tua proprietà nella Guida per l'IAMutente.

- Per scoprire come fornire l'accesso alle tue risorse a terze parti Account AWS, consulta Fornire l'accesso a persone Account AWS di proprietà di terzi nella Guida per l'IAMutente.
- Per informazioni su come fornire l'accesso tramite la federazione delle identità, consulta Fornire l'accesso agli utenti autenticati esternamente (federazione delle identità) nella Guida per l'IAMutente.
- Per conoscere la differenza tra l'utilizzo di ruoli e politiche basate sulle risorse per l'accesso tra account diversi, consulta la sezione Accesso alle risorse tra account nella Guida per l'utente. IAM IAM

Utilizzo dei ruoli collegati ai servizi per API Gateway

Amazon API Gateway utilizza AWS Identity and Access Management ruoli collegati ai servizi (IAM). Un ruolo collegato ai servizi è un tipo di ruolo specifico di IAM collegato direttamente ad API Gateway. I ruoli collegati ai servizi sono predefiniti da API Gateway e includono tutte le autorizzazioni richieste dal servizio per chiamare altri AWS servizi per tuo conto.

Un ruolo collegato ai servizi semplifica la configurazione di API Gateway perché permette di evitare l'aggiunta manuale delle autorizzazioni necessarie. API Gateway definisce le autorizzazioni dei relativi ruoli associati ai servizi e, salvo diversamente definito, solo API Gateway potrà assumere i propri ruoli. Le autorizzazioni definite includono la policy di attendibilità e la policy delle autorizzazioni che non può essere collegata a nessun'altra entità IAM.

È possibile eliminare un ruolo collegato ai servizi solo dopo aver eliminato le risorse correlate. Questa procedura protegge le risorse di API Gateway perché impedisce la rimozione involontaria delle autorizzazioni di accesso alle risorse.

Per informazioni su altri servizi che supportano i ruoli collegati ai servizi, consulta AWS Services That Work with IAM e cerca i servizi che riportano Yes (Sì) nella colonna Service-Linked Role (Ruolo associato ai servizi). Scegli un link Yes (Sì) per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

Autorizzazioni del ruolo collegato ai servizi per API Gateway

API Gateway utilizza il ruolo collegato al servizio denominato AWSServiceRoleForAPIGateway—Consente ad API Gateway di accedere a Elastic Load Balancing, Amazon Data Firehose e altre risorse di servizio per tuo conto.

Il ruolo AWSServiceRoleForAPIGateway collegato al servizio prevede che i seguenti servizi assumano il ruolo:

- `ops.apigateway.amazonaws.com`

La policy delle autorizzazioni del ruolo consente ad API Gateway di eseguire le seguenti operazioni sulle risorse specificate:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:AddListenerCertificates",
                "elasticloadbalancing:RemoveListenerCertificates",
                "elasticloadbalancing:ModifyListener",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:DescribeLoadBalancers",
                "xray:PutTraceSegments",
                "xray:PutTelemetryRecords",
                "xray:GetSamplingTargets",
                "xray:GetSamplingRules",
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries",
                "servicediscovery:DiscoverInstances"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "firehose:DescribeDeliveryStream",
                "firehose:PutRecord",
                "firehose:PutRecordBatch"
            ],
            "Resource": "arn:aws:firehose:*:*:deliverystream/amazon-apigateway-*"
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "acm:DescribeCertificate",
                "acm:GetCertificate"
            ],
            "Resource": "arn:aws:acm:*:*:certificate/*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateNetworkInterfacePermission",
            "Resource": "arn:aws:ec2:*:*:network-interface/*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:*:*:network-interface/*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": [
                        "Owner",
                        "VpcLinkId"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:DeleteNetworkInterface",
                "ec2:AssignPrivateIpAddresses",
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeNetworkInterfaceAttribute",
                "ec2:DescribeVpcs",
                "ec2:DescribeNetworkInterfacePermissions",
                "ec2:UnassignPrivateIpAddresses",
                "ec2:DescribeSubnets",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups"
            ],
            "Resource": "*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": "servicediscovery:GetNamespace",
            "Resource": "arn:aws:servicediscovery:*:*:namespace/*"
        },
        {
            "Effect": "Allow",
            "Action": "servicediscovery:GetService",
            "Resource": "arn:aws:servicediscovery:*:*:service/*"
        }
    ]
}
```

Per consentire a un'entità IAM (come un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato ai servizi devi configurare le relative autorizzazioni. Per ulteriori informazioni, consulta Autorizzazioni del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Creazione di un ruolo collegato ai servizi per API Gateway

Non hai bisogno di creare manualmente un ruolo collegato ai servizi. Quando crei un'API, un nome di dominio personalizzato o un link VPC nell' AWS API AWS Management Console, API Gateway crea automaticamente il ruolo collegato al servizio. AWS CLI

Se elimini questo ruolo collegato ai servizi, puoi ricrearlo seguendo lo stesso processo utilizzato per ricreare il ruolo nell'account. Quando si crea un'API, un nome di dominio personalizzato o un collegamento VPC, API Gateway crea nuovamente il ruolo collegato ai servizi per l'utente.

Modifica di un ruolo collegato ai servizi per API Gateway

API Gateway non consente di modificare il ruolo AWSServiceRoleForAPIGateway collegato al servizio. Dopo aver creato un ruolo collegato al servizio, non puoi modificarne il nome, perché potrebbero farvi riferimento diverse entità. Puoi tuttavia modificarne la descrizione utilizzando IAM. Per ulteriori informazioni, consulta la sezione Modifica di un ruolo collegato ai servizi nella Guida per l'utente di IAM.

Eliminazione di un ruolo collegato ai servizi per API Gateway

Se non è più necessario utilizzare una caratteristica o un servizio che richiede un ruolo collegato ai servizi, ti consigliamo di eliminare quel ruolo. In questo modo non sarà più presente un'entità non utilizzata che non viene monitorata e gestita attivamente. Tuttavia, è necessario effettuare la pulizia delle risorse associate al ruolo collegato ai servizi prima di poterlo eliminare manualmente.

> **Note**
>
> Se il servizio API Gateway utilizza tale ruolo quando tenti di eliminare le risorse, è possibile che l'eliminazione non abbia esito positivo. In questo caso, attendi alcuni minuti e quindi ripeti l'operazione.

Per eliminare le risorse API Gateway utilizzate da AWSServiceRoleForAPIGateway

1. Aprire la console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway/.
2. Passare all'API, al nome di dominio personalizzato o al collegamento VPC che utilizza il ruolo collegato al servizio.
3. Utilizza la console per eliminare una risorsa.
4. Ripetere la procedura per eliminare tutte le API, i nomi di dominio personalizzati o i collegamenti VPC che utilizzano il ruolo collegato al servizio.

Per eliminare manualmente il ruolo collegato ai servizi mediante IAM

Utilizza la console IAM AWS CLI, o l' AWS API per eliminare il ruolo AWSServiceRoleForAPIGateway collegato al servizio. Per ulteriori informazioni, consulta Eliminazione del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Regioni supportate per i ruoli collegati ai servizi di API Gateway

API Gateway supporta l'utilizzo di ruoli collegati ai servizi in tutte le regioni in cui il servizio è disponibile. Per ulteriori informazioni, consulta AWS Endpoint del servizio.

Aggiornamenti API Gateway alle policy AWS gestite

Visualizza i dettagli sugli aggiornamenti delle politiche AWS gestite per API Gateway da quando questo servizio ha iniziato a tenere traccia di queste modifiche. Per gli avvisi automatici sulle modifiche apportate a questa pagina, sottoscrivere il feed RSS nella pagina della cronologia dei documenti di API Gateway.

Modifica	Descrizione	Data
Aggiunto supporto `acm:GetCertificate`	La policy `AWSServiceRoleForAPIGateway`	12 luglio 2021

Modifica	Descrizione	Data
per la policy `AWSServiceRoleForAPIGateway` .	ora include l'autorizzazione per chiamare l'operazione API `GetCertificate` ACM.	
API Gateway ha iniziato a monitorare le modifiche	API Gateway ha iniziato a tracciare le modifiche per le sue politiche AWS gestite.	12 luglio 2021

Logging e monitoraggio in Amazon API Gateway

Il monitoraggio è una parte importante per mantenere l'affidabilità, la disponibilità e le prestazioni di API Gateway e delle tue AWS soluzioni. È necessario raccogliere i dati di monitoraggio da tutte le parti della AWS soluzione in modo da poter eseguire più facilmente il debug di un errore multipunto, se si verifica. AWS fornisce diversi strumenti per monitorare le risorse dell'API Gateway e rispondere a potenziali incidenti:

CloudWatch Registri Amazon

Per risolvere i problemi relativi all'esecuzione delle richieste o all'accesso del client alla tua API, puoi abilitare CloudWatch Logs per registrare le chiamate API. Per ulteriori informazioni, consulta the section called “CloudWatch registri”.

CloudWatch Allarmi Amazon

Utilizzando gli CloudWatch allarmi, controlli una singola metrica per un periodo di tempo specificato. Se la metrica supera una determinata soglia, viene inviata una notifica a un argomento o AWS Auto Scaling una politica di Amazon Simple Notification Service. CloudWatch gli allarmi non richiamano azioni quando una metrica si trova in uno stato particolare. È necessario invece cambiare lo stato e mantenerlo per un numero di periodi specificato. Per ulteriori informazioni, consulta the section called “CloudWatch metriche”.

Registrazione degli accessi a Firehose

Per facilitare il debug dei problemi relativi all'accesso del client all'API, è possibile abilitare Firehose per registrare le chiamate API. Per ulteriori informazioni, consulta the section called “Firehose”.

AWS CloudTrail

CloudTrail fornisce un registro delle azioni intraprese da un utente, un ruolo o un AWS servizio in API Gateway. Utilizzando le informazioni raccolte da CloudTrail, è possibile determinare la richiesta effettuata a API Gateway, 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, consulta the section called “Lavorare con CloudTrail”.

AWS X-Ray

X-Ray è un AWS servizio che raccoglie dati sulle richieste servite dall'applicazione e li utilizza per creare una mappa dei servizi che è possibile utilizzare per identificare problemi con l'applicazione e opportunità di ottimizzazione. Per ulteriori informazioni, consulta the section called “Configurare AWS X-Ray”.

AWS Config

AWS Config fornisce una visualizzazione dettagliata della configurazione delle AWS risorse dell'account. Puoi vedere in che modo le risorse sono correlate, ottenere una cronologia delle modifiche alla configurazione ed esaminare come cambiano le relazioni e le configurazioni nel tempo. È possibile utilizzare AWS Config per definire regole che valutano le configurazioni delle risorse per la conformità dei dati. AWS Config le regole rappresentano le impostazioni di configurazione ideali per le risorse API Gateway. Se una risorsa viola una regola e viene contrassegnata come non conforme, può AWS Config avvisarti utilizzando un argomento di Amazon Simple Notification Service (Amazon SNS). Per informazioni dettagliate, vedi the section called “Lavorare con AWS Config”.

Registrazione delle chiamate API di Amazon API Gateway utilizzando AWS CloudTrail

Amazon API Gateway è integrato con AWS CloudTrail, un servizio che fornisce un registro delle azioni intraprese da un utente, ruolo o un Servizio AWS. CloudTrail acquisisce tutte le chiamate API REST per il servizio API Gateway come eventi. Le chiamate acquisite includono chiamate dalla console API Gateway e chiamate di codice alle API del servizio API Gateway. Utilizzando le informazioni raccolte da CloudTrail, è possibile determinare la richiesta effettuata a API Gateway, l'indirizzo IP da cui è stata effettuata la richiesta, quando è stata effettuata e dettagli aggiuntivi.

> **Note**
>
> TestInvokeAuthorizere non TestInvokeMethodhanno effettuato l'accesso. CloudTrail

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 le credenziali utente root o utente.
- Se la richiesta è stata effettuata per conto di un utente IAM Identity Center.
- 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 Servizio AWS.

CloudTrail è attivo nel tuo account Account AWS quando crei l'account e hai automaticamente accesso alla cronologia degli CloudTrail eventi. La cronologia CloudTrail degli eventi fornisce un record visualizzabile, ricercabile, scaricabile e immutabile degli ultimi 90 giorni di eventi di gestione registrati in un. Regione AWSPer ulteriori informazioni, consulta Lavorare con la cronologia degli CloudTrail eventi nella Guida per l'utente.AWS CloudTrail Non sono CloudTrail previsti costi per la visualizzazione della cronologia degli eventi.

Per una registrazione continua degli eventi degli Account AWS ultimi 90 giorni, crea un trail o un data store di eventi CloudTrailLake.

CloudTrail sentieri

Un trail consente di CloudTrail inviare file di log a un bucket Amazon S3. Tutti i percorsi creati utilizzando il AWS Management Console sono multiregionali. È possibile creare un percorso a regione singola o multiregione utilizzando. AWS CLI La creazione di un percorso multiregionale è consigliata in quanto consente di registrare l'intera attività del proprio account Regioni AWS . Se crei un percorso a regione singola, puoi visualizzare solo gli eventi registrati nel percorso. Regione AWS Per ulteriori informazioni sui percorsi, consulta Creazione di un percorso per te Account AWS e Creazione di un percorso per un'organizzazione nella Guida per l'AWS CloudTrail utente.

Puoi inviare gratuitamente una copia dei tuoi eventi di gestione in corso al tuo bucket Amazon S3 CloudTrail creando un percorso, tuttavia ci sono costi di storage di Amazon S3. Per ulteriori informazioni sui CloudTrail prezzi, consulta la pagina Prezzi.AWS CloudTrail Per informazioni sui prezzi di Amazon S3, consulta Prezzi di Amazon S3.

CloudTrail Archivi di dati sugli eventi di Lake

CloudTrail Lake ti consente di eseguire query basate su SQL sui tuoi eventi. CloudTrail Lake converte gli eventi esistenti in formato JSON basato su righe in formato Apache ORC. ORC è un formato di archiviazione a colonne ottimizzato per il recupero rapido dei dati. Gli eventi vengono aggregati in archivi di dati degli eventi, che sono raccolte di eventi immutabili basate sui criteri selezionati applicando i selettori di eventi avanzati. I selettori applicati a un archivio di dati degli eventi controllano quali eventi persistono e sono disponibili per l'esecuzione della query. Per ulteriori informazioni su CloudTrail Lake, consulta Working with AWS CloudTrail Lake nella Guida per l'utente.AWS CloudTrail

CloudTrail Gli archivi e le richieste di dati sugli eventi di Lake comportano dei costi. Quando crei un datastore di eventi, scegli l'opzione di prezzo da utilizzare per tale datastore. L'opzione di prezzo determina il costo per l'importazione e l'archiviazione degli eventi, nonché il periodo di conservazione predefinito e quello massimo per il datastore di eventi. Per ulteriori informazioni sui CloudTrail prezzi, consulta Prezzi.AWS CloudTrail

Eventi di gestione dell'API Gateway in CloudTrail

Gli eventi di gestione forniscono informazioni sulle operazioni di gestione eseguite sulle risorse di Account AWS. Queste operazioni sono definite anche operazioni del piano di controllo (control-plane). Per impostazione predefinita, CloudTrail registra gli eventi di gestione.

Amazon API Gateway registra tutte le azioni di API Gateway come eventi di gestione, ad eccezione di TestInvokeAuthorizere TestInvokeMethod. Per un elenco delle azioni di Amazon API Gateway a cui API Gateway effettua il log CloudTrail, consulta l'Amazon API Gateway API Reference.

Esempio di evento API Gateway

Un evento rappresenta una singola richiesta proveniente da qualsiasi fonte e include informazioni sull'operazione API richiesta, la data e l'ora dell'operazione, i parametri della richiesta e così via. CloudTrail i file di registro non sono una traccia ordinata dello stack delle chiamate API pubbliche, quindi gli eventi non vengono visualizzati in un ordine specifico.

L'esempio seguente mostra un CloudTrail evento che dimostra l'`GetResource`azione API Gateway:

```
{
    Records: [
        {
```

```
            eventVersion: "1.03",
            userIdentity: {
                type: "Root",
                principalId: "AKIAI44QH8DHBEXAMPLE",
                arn: "arn:aws:iam::123456789012:root",
                accountId: "123456789012",
                accessKeyId: "AKIAIOSFODNN7EXAMPLE",
                sessionContext: {
                    attributes: {
                        mfaAuthenticated: "false",
                        creationDate: "2015-06-16T23:37:58Z"
                    }
                }
            },
            eventTime: "2015-06-17T00:47:28Z",
            eventSource: "apigateway.amazonaws.com",
            eventName: "GetResource",
            awsRegion: "us-east-1",
            sourceIPAddress: "203.0.113.11",
            userAgent: "example-user-agent-string",
            requestParameters: {
                restApiId: "3rbEXAMPLE",
                resourceId: "5tfEXAMPLE",
                template: false
            },
            responseElements: null,
            requestID: "6d9c4bfc-148a-11e5-81b6-7577cEXAMPLE",
            eventID: "4d293154-a15b-4c33-9e0a-ff5eeEXAMPLE",
            readOnly: true,
            eventType: "AwsApiCall",
            recipientAccountId: "123456789012"
        },
        ... additional entries ...
    ]
}
```

Per informazioni sul contenuto dei CloudTrail record, consulta il contenuto dei CloudTrail record nella Guida per l'AWS CloudTrail utente.

Monitoraggio della configurazione API di API Gateway con AWS Config

Puoi usare AWS Config per registrare le modifiche di configurazione apportate alle risorse API Gateway API e inviare notifiche in base alle modifiche alle risorse. Conservare la cronologia delle

modifiche eseguite sulla configurazione delle risorse di API Gateway è utile per la risoluzione dei problemi operativi, la revisione e i casi d'uso relativi alla conformità.

AWS Config è in grado di monitorare le modifiche apportate a:

- Configurazione delle fasi API, ad esempio:
  - Impostazioni del cluster di cache
  - Impostazioni di throttling
  - Impostazioni dei log degli accessi
  - Set di distribuzione attivo nella fase
- Configurazione dell'API, ad esempio:
  - Configurazione dell'endpoint
  - Versione
  - protocol
  - tags

Inoltre, la funzionalità Regole di AWS Config permette di definire regole di configurazione e rilevare, monitorare e comunicare tramite avvisi automaticamente eventuali violazioni a queste regole. La funzionalità di monitoraggio delle modifiche apportate alle proprietà di configurazione della risorsa consente, inoltre, di creare regole AWS Config attivate da modifiche per le risorse API Gateway e verificarne le configurazioni rispetto alle best practice.

Puoi abilitare AWS Config nell'account usando la console AWS Config o AWS CLI. Seleziona i tipi di risorsa per cui vuoi monitorare le modifiche. Se in precedenza hai configurato la registrazione di tutti i tipi di risorse su AWS Config, anche quelle di API Gateway verranno automaticamente registrate nel tuo account. Il supporto per Amazon API Gateway, nell'ambito del servizio AWS Config, è disponibile in tutte le Regioni pubbliche AWS e AWS GovCloud (US). Per l'elenco completo delle Regioni supportate, consulta Endpoint e quote Amazon API Gateway nei riferimenti generali Riferimenti generali di AWS.

Argomenti

- Tipi di risorse supportati
- Configurazione di AWS Config
- Configurazione di AWS Config per la registrazione delle risorse API Gateway
- Visualizzazione dei dettagli di configurazione di API Gateway nella console AWS Config

- Valutazione delle risorse di API Gateway utilizzando le regole di AWS Config

Tipi di risorse supportati

I tipi di risorsa API Gateway seguenti sono integrati con AWS Config e vengono descritti in Tipi di risorse e relazioni tra risorse AWS supportati da AWS Config:

- `AWS::ApiGatewayV2::Api` (API WebSocket e HTTP)
- `AWS::ApiGateway::RestApi` (API REST)
- `AWS::ApiGatewayV2::Stage` (fase API WebSocket e HTTP)
- `AWS::ApiGateway::Stage` (fase API REST)

Per ulteriori informazioni su AWS Config, consulta la Guida per lo sviluppatore di AWS Config. Per informazioni sui prezzi, consulta la pagina di informazioni sui prezzi di AWS Config.

> **Important**
>
> Se modifichi una delle proprietà dell'API seguenti dopo avere distribuito l'API, devi ridistribuire l'API per propagare le modifiche. In caso contrario, le modifiche agli attributi verranno visualizzate nella console AWS Config, ma le impostazioni delle proprietà precedenti saranno ancora valide. il comportamento di runtime dell'API resterà invariato.
>
> - **`AWS::ApiGateway::RestApi`** – `binaryMediaTypes`, `minimumCompressionSize`, `apiKeySource`
> - **`AWS::ApiGatewayV2::Api`** – `apiKeySelectionExpression`

Configurazione di AWS Config

Per la configurazione iniziale di AWS Config, consulta gli argomenti seguenti nella Guida per gli sviluppatori di AWS Config.

- Configurazione di AWS Config con la console
- Configurazione di AWS Config con AWS CLI

Configurazione di AWS Config per la registrazione delle risorse API Gateway

Per impostazione predefinita, AWS Config registra le modifiche di configurazione per tutti i tipi supportati di risorse regionali individuati nella regione in cui è in esecuzione l'ambiente. Puoi personalizzare AWS Config in modo da registrare le modifiche solo per tipi di risorse specifici oppure per le risorse globali.

Per informazioni sulle differenze tra risorse regionali e globali e su come personalizzare la configurazione di AWS Config, consulta la pagina relativa alla selezione delle risorse registrate da AWS Config.

Visualizzazione dei dettagli di configurazione di API Gateway nella console AWS Config

Puoi usare la console AWS Config per cercare le risorse API Gateway e ottenere informazioni correnti e cronologiche sulle loro configurazioni. La procedura seguente descrive come trovare informazioni su un'API di API Gateway.

Per trovare una risorsa API Gateway nella console di configurazione AWS

1. Aprire la console AWS Config.
2. Scegliere Resources (Risorse).
3. Nella pagina Resource inventory (Inventario risorse), scegliere Resources (Risorse).
4. Aprire il menu Resource type (Tipo di risorsa), scorrere fino ad APIGateway o APIGatewayV2 e quindi scegliere uno o più dei tipi di risorsa API Gateway.
5. Scegliere Look up (Cerca).
6. Scegliere un ID risorsa nell'elenco delle risorse visualizzate da AWS Config. AWS Config mostra i dettagli di configurazione e altre informazioni sulla risorsa selezionata.
7. Per visualizzare i dettagli completi della configurazione registrata, scegliere View Details (Visualizza dettagli).

Per ulteriori informazioni su come trovare una risorsa e visualizzare le informazioni in questa pagina, consulta Visualizzazione delle configurazioni e della cronologia delle risorseAWS nella Guida per gli sviluppatori di AWS Config.

Valutazione delle risorse di API Gateway utilizzando le regole di AWS Config

Puoi creare regole di AWS Config, che rappresentano le impostazioni di configurazione ideali per le risorse API Gateway. Puoi usare regole gestite AWS Config predefinite oppure definire regole personalizzate. AWS Config monitora continuamente le modifiche apportate alla configurazione delle risorse per determinare se violino una o più delle condizioni nelle regole. La console AWS Config mostra lo stato di conformità delle regole e delle risorse.

Se una risorsa viola una regola e viene contrassegnata come non conforme, AWS Config può avvisare l'utente utilizzando un argomento Guida per gli sviluppatori di Amazon Simple Notification Service (Amazon SNS). Per utilizzare i dati a livello di programmazione in questi avvisi AWS Config, usa una coda Amazon Simple Queue Service (Amazon SQS) come l'endpoint di notifica per l'argomento Amazon SNS.

Per ulteriori informazioni sulla configurazione e l'uso di regole, consulta Valutazione delle risorse con le regole nella Guida per gli sviluppatori di AWS Config.

Convalida della conformità per Amazon API Gateway

Per sapere se un Servizio AWS programma rientra nell'ambito di specifici programmi di conformità, consulta Servizi AWS la sezione Scope by Compliance Program Servizi AWS e scegli il programma di conformità che ti interessa. Per informazioni generali, consulta Programmi di AWS conformità Programmi di di .

È possibile scaricare report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta Scaricamento dei report in AWS Artifact .

La vostra responsabilità di conformità durante l'utilizzo Servizi AWS è determinata dalla sensibilità dei dati, dagli obiettivi di conformità dell'azienda e dalle leggi e dai regolamenti applicabili. AWS fornisce le seguenti risorse per contribuire alla conformità:

- Guide introduttive su sicurezza e conformità: queste guide all'implementazione illustrano considerazioni sull'architettura e forniscono passaggi per implementare ambienti di base incentrati sulla AWS sicurezza e la conformità.
- Architettura per la HIPAA sicurezza e la conformità su Amazon Web Services: questo white paper descrive in che modo le aziende possono utilizzare AWS per creare applicazioni idonee. HIPAA

> **Note**
>
> Non tutte sono idonee. Servizi AWS HIPAA Per ulteriori informazioni, consulta la Guida ai servizi HIPAA idonei.

- AWS Risorse per AWS per la conformità: questa raccolta di cartelle di lavoro e guide potrebbe riguardare il tuo settore e la tua località.
- AWS Guide 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
- Evaluating Resources with Rules nella Guida per gli AWS Config sviluppatori: il AWS Config servizio valuta la conformità delle configurazioni delle risorse alle pratiche interne, alle linee guida del settore e alle normative.
- AWS Security Hub— Ciò Servizio AWS fornisce una visione completa dello stato di sicurezza interno. AWS La Centrale di sicurezza utilizza i controlli di sicurezza per valutare le risorse AWS e 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.
- Amazon GuardDuty: Servizio AWS rileva potenziali minacce ai tuoi carichi di lavoro Account AWS, ai contenitori e ai dati monitorando l'ambiente alla ricerca di attività sospette e dannose. GuardDuty può aiutarti a soddisfare vari requisiti di conformità, ad esempio PCI DSS soddisfacendo i requisiti di rilevamento delle intrusioni imposti da determinati framework di conformità.
- AWS Audit Manager— Ciò Servizio AWS consente di verificare continuamente AWS l'utilizzo per semplificare la gestione del rischio e la conformità alle normative e agli standard di settore.

Resilienza in Amazon API Gateway

L'infrastruttura AWS globale è costruita attorno a AWS regioni e zone di disponibilità. AWS Le regioni forniscono più zone di disponibilità fisicamente separate e isolate, collegate con reti a bassa latenza, ad alto throughput e altamente ridondanti. Con le zone di disponibilità, puoi progettare e gestire applicazioni e database che eseguono automaticamente il failover 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 tradizionali.

Essendo un servizio regionale completamente gestito, API Gateway opera in più zone di disponibilità in ciascuna regione, utilizzando la ridondanza delle zone di disponibilità per ridurre al minimo i guasti dell'infrastruttura come categoria di rischio di disponibilità. API Gateway è progettato per il ripristino automatico in caso di guasto di una zona di disponibilità.

Per ulteriori informazioni su AWS regioni e zone di disponibilità, consulta AWS Global Infrastructure.

Per evitare che le API vengano sopraffatte da troppe richieste, API Gateway limita le richieste alle API. Nello specifico, API Gateway imposta un limite per il numero di richieste durante il funzionamento a regime e per i picchi di invio di richieste per tutte le API dell'account. È possibile configurare la limitazione personalizzata per le API. Per ulteriori informazioni, consulta Limita le richieste alle tue API REST per una migliore velocità di trasmissione in API Gateway.

È possibile utilizzare i controlli dell'integrità di Route 53 per controllare il failover DNS da un'API di API Gateway in una regione primaria a un'API di Gateway API in un'area secondaria. Per un esempio, consulta the section called “DNSfailover”.

Sicurezza dell'infrastruttura in Amazon API Gateway

In quanto servizio gestito, Amazon API Gateway è protetto dalla sicurezza di rete AWS globale. Per informazioni sui servizi AWS di sicurezza e su come AWS protegge l'infrastruttura, consulta AWS Cloud Security. Per progettare il tuo AWS ambiente utilizzando le migliori pratiche per la sicurezza dell'infrastruttura, vedi Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Si utilizzano API chiamate AWS pubblicate per accedere a API Gateway attraverso la rete. I client devono supportare quanto segue:

- Transport Layer Security (TLS). Richiediamo TLS 1.2 e consigliamo TLS 1.3.
- Suite di cifratura con Perfect Forward Secrecy (PFS) come (Ephemeral Diffie-Hellman) o DHE (Elliptic Curve Ephemeral Diffie-Hellman). ECDHE La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Inoltre, le richieste devono essere firmate utilizzando un ID chiave di accesso e una chiave di accesso segreta associata a un principale. IAM O puoi utilizzare AWS Security Token Service (AWS STS) per generare credenziali di sicurezza temporanee per sottoscrivere le richieste.

È possibile richiamare queste API operazioni da qualsiasi posizione di rete, ma API Gateway supporta politiche di accesso basate sulle risorse, che possono includere restrizioni basate

sull'indirizzo IP di origine. Puoi anche utilizzare policy basate sulle risorse per controllare l'accesso da endpoint Amazon Virtual Private Cloud (AmazonVPC) specifici o specifici. VPCs In effetti, questo isola l'accesso alla rete a una determinata risorsa API Gateway solo da quella specifica all'interno della rete. VPC AWS

Analisi delle vulnerabilità in Amazon API Gateway

Configurazione e controllo IT sono una responsabilità condivisa tra AWS e te, il nostro cliente. Per ulteriori informazioni, consulta il modello di responsabilità condivisa di AWS.

Best practice di sicurezza in Amazon API Gateway

API Gateway fornisce una serie di caratteristiche di sicurezza che occorre valutare durante lo sviluppo e l'implementazione delle policy di sicurezza. Le seguenti best practice sono linee guida generali e non rappresentano una soluzione di sicurezza completa. Poiché queste best practice potrebbero non essere appropriate o sufficienti per l'ambiente, gestiscile come considerazioni utili anziché prescrizioni.

Implementazione dell'accesso con privilegi minimi

Utilizzare le policy IAM per implementare l'accesso con privilegi minimi per la creazione, la lettura, l'aggiornamento o l'eliminazione delle API di API Gateway. Per ulteriori informazioni, consulta Gestione delle identità e degli accessi per Amazon API Gateway. API Gateway offre diverse opzioni per controllare l'accesso alle API create. Per ulteriori informazioni, consulta Controlla e gestisci l'accesso alle API REST in API Gateway, Controlla e gestisci l'accesso alle WebSocket API in API Gateway e Controlla l'accesso HTTP APIs con gli JWT autorizzatori in API Gateway.

Implementare la registrazione

Usa CloudWatch Logs o Amazon Data Firehose per registrare le richieste alle tue API. Per ulteriori informazioni, consulta Monitora le API REST in API Gateway, Configurare la registrazione per WebSocket APIs Gateway API e Configurare la registrazione per le API HTTP in API Gateway.

Implementa gli CloudWatch allarmi Amazon

Utilizzando gli CloudWatch allarmi, controlli una singola metrica per un periodo di tempo specificato. Se la metrica supera una determinata soglia, viene inviata una notifica a un argomento o AWS Auto Scaling una politica di Amazon Simple Notification Service. CloudWatch gli allarmi non richiamano azioni quando una metrica si trova in uno stato particolare. È

necessario invece cambiare lo stato e mantenerlo per un numero di periodi specificato. Per ulteriori informazioni, consulta the section called “CloudWatch metriche”.

Abilita AWS CloudTrail

CloudTrail fornisce un registro delle azioni intraprese da un utente, un ruolo o un AWS servizio in API Gateway. Utilizzando le informazioni raccolte da CloudTrail, è possibile determinare la richiesta effettuata a API Gateway, 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, consulta the section called “Lavorare con CloudTrail”.

Abilita AWS Config

AWS Config fornisce una visualizzazione dettagliata della configurazione delle AWS risorse del tuo account. Puoi vedere in che modo le risorse sono correlate, ottenere una cronologia delle modifiche alla configurazione ed esaminare come cambiano le relazioni e le configurazioni nel tempo. È possibile utilizzare AWS Config per definire regole che valutano le configurazioni delle risorse per la conformità dei dati. AWS Config le regole rappresentano le impostazioni di configurazione ideali per le risorse API Gateway. Se una risorsa viola una regola e viene contrassegnata come non conforme, può AWS Config avvisarti utilizzando un argomento di Amazon Simple Notification Service (Amazon SNS). Per informazioni dettagliate, vedi the section called “Lavorare con AWS Config”.

Usa AWS Security Hub

Monitora l'utilizzo di API Gateway riguardo alle best practice di sicurezza utilizzando AWS Security Hub. Security Hub utilizza controlli di sicurezza per valutare le configurazioni delle risorse e gli standard di sicurezza per aiutarti a rispettare vari framework di conformità. Per ulteriori informazioni sull'utilizzo di Security Hub volto a valutare le risorse Lambda, consulta i controlli di Amazon API Gateway nella Guida per l'utente di AWS Security Hub .

Tagging delle risorse API Gateway

Un tag è un'etichetta di metadati che si assegna o che si AWS assegna a una risorsa. AWS Ogni tag è costituito da due parti:

- Una chiave del tag (ad esempio, `CostCenter`, `Environment` o `Project`). Le chiavi dei tag prevedono una distinzione tra lettere maiuscole e minuscole.
- Un campo facoltativo noto come valore del tag (ad esempio, `111122223333` o `Production`). Non specificare il valore del tag equivale a utilizzare una stringa vuota. Analogamente alle chiavi dei tag, i valori dei tag rilevano la distinzione tra maiuscole e minuscole.

I tag consentono di eseguire le seguenti operazioni:

- Controllare gli accessi alle risorse in base ai tag a loro assegnati. Puoi controllare gli accessi specificando chiavi e valori di tag nelle condizioni di una policy AWS Identity and Access Management (IAM). Per ulteriori informazioni sul controllo degli accessi basato su tag, consulta l'argomento relativo al controllo degli accessi mediante tag nella Guida per l'utente di IAM.
- Tieni traccia dei costi. AWS Attivi questi tag sulla AWS Billing and Cost Management dashboard. AWS utilizza i tag per classificare i costi e fornirti un rapporto mensile sull'allocazione dei costi. Per ulteriori informazioni, consulta la pagina sull'utilizzo dei tag per l'allocazione dei costi nella Guida per l'utente di AWS Billing.
- Identifica e organizza le tue risorse. AWS Molti AWS servizi supportano l'etichettatura, quindi puoi assegnare lo stesso tag a risorse di servizi diversi per indicare che le risorse sono correlate. Ad esempio, è possibile assegnare a uno stadio API Gateway lo stesso tag assegnato a una regola CloudWatch Events.

Per suggerimenti sull'uso dei tag, consulta il white paper AWS Tagging Strategies.

Nelle sezioni seguenti vengono fornite ulteriori informazioni sui tag di Amazon API Gateway.

Argomenti

- Risorse API Gateway a cui è possibile applicare tag
- Utilizzo dei tag per controllare l'accesso alle risorse REST API di Gateway API

Risorse API Gateway a cui è possibile applicare tag

I tag possono essere impostati sulle seguenti risorse API HTTP o WebSocket API nell'API Amazon API Gateway V2:

- `Api`
- `DomainName`
- `Stage`
- `VpcLink`

Inoltre, i tag possono essere impostati sulle seguenti risorse di API REST nell'API di Amazon API Gateway V1:

- `ApiKey`
- `ClientCertificate`
- `DomainName`
- `RestApi`
- `Stage`
- `UsagePlan`
- `VpcLink`

I tag non possono essere impostati direttamente su altre risorse. Tuttavia, nell'API di Amazon API Gateway V1, le risorse figlio ereditano i tag impostati sulle risorse padre. Per esempio:

- Se un tag è impostato su una risorsa `RestApi`, questo tag è ereditato dalle seguenti risorse figlio di tale `RestApi` per il Controllo degli accessi basato sugli attributi:
  - `Authorizer`
  - `Deployment`
  - `Documentation`
  - `GatewayResponse`
  - `Integration`
  - `Method`
  - `Model`

  - `Resource`
  - `ResourcePolicy`
  - `Setting`
  - `Stage`
- Se un tag è impostato su un `DomainName`, tale tag è ereditato da qualsiasi risorsa `BasePathMapping` al di sotto di esso.
- Se un tag è impostato su un `UsagePlan`, tale tag è ereditato da qualsiasi risorsa `UsagePlanKey` al di sotto di esso.

> **Note**
>
> L'eredità dei tag viene applicata solo per il controllo degli accessi basato sugli attributi. Ad esempio, non puoi utilizzare tag ereditati per monitorare i costi in. AWS Cost Explorer Per utilizzare i tag per l'allocazione dei costi, ti consigliamo di creare tag sulle risorse secondarie, come la `Stage` risorsa.
> API Gateway non restituisce tag ereditati quando GetTagsrichiedi una risorsa.

Eredità dei tag nell'API di Amazon API Gateway V1

In precedenza, era solo possibile impostare tag su fasi. Ora che è possibile impostarli anche su altre risorse, uno `Stage` può ricevere un tag in due modi:

- Il tag può essere impostato direttamente sullo `Stage`.
- La fase può ereditare il tag dalla sua padr `RestApi`.

Se una fase riceve un tag in entrambi i modi, il tag che è stato impostato direttamente sulla fase ha la precedenza. Ad esempio, supponiamo che una fase erediti i seguenti tag dalla sua API REST padre:

```
{
 'foo': 'bar',
 'x':'y'
}
```

Supponiamo anche che disponga dei seguenti tag impostati direttamente su di essa:

```
{
 'foo': 'bar2',
 'hello': 'world'
}
```

Il risultato netto è che la fase dispone dei seguenti tag, con i seguenti valori:

```
{
 'foo': 'bar2',
 'hello': 'world'
 'x':'y'
}
```

Limitazioni applicate ai tag e convenzioni di utilizzo

Le seguenti limitazioni e convenzioni di utilizzo si applicano all'utilizzo dei tag con risorse API Gateway:

- Ogni risorsa può avere un massimo di 50 tag.
- Per ciascuna risorsa, ogni chiave del tag deve essere univoca e ogni chiave del tag può avere un solo valore.
- La lunghezza massima delle chiavi di tag è 128 caratteri Unicode in UTF-8.
- Il valore massimo dei tag è 256 caratteri Unicode in UTF-8.
- I caratteri consentiti per chiavi e valori sono lettere, numeri, spazi rappresentabili in formato UTF-8, oltre ai seguenti caratteri: . : + = @ _ / - (trattino). Le risorse Amazon EC2 consentono qualsiasi carattere.
- Per chiavi e valori di tag viene fatta la distinzione tra maiuscole e minuscole. Come best practice, è consigliabile definire una strategia per l'uso delle lettere maiuscole e minuscole nei tag e implementarla costantemente in tutti i tipi di risorse. Ad esempio, puoi decidere se utilizzare `Costcenter`, `costcenter` o `CostCenter` e utilizzare la stessa convenzione per tutti i tag. Non utilizzare tag simili con lettere maiuscole o minuscole incoerenti.
- Il prefisso `aws:` non può essere utilizzato con i tag; è riservato per l'uso in AWS . Non è possibile modificare né eliminare le chiavi o i valori di tag con tale prefisso. I tag con questo prefisso non vengono conteggiati per il limite del numero di tag per risorsa.

Utilizzo dei tag per controllare l'accesso alle risorse REST API di Gateway API

Le condizioni nelle AWS Identity and Access Management policy fanno parte della sintassi utilizzata per specificare le autorizzazioni per le risorse API Gateway. Per ulteriori informazioni su come specificare le policy IAM, consulta the section called “Usa le IAM autorizzazioni”. In API Gateway, le risorse possono avere tag e alcune operazioni possono includere tag. Quando si crea una policy IAM, è possibile utilizzare le chiavi di condizione di tag per controllare:

- Quali utenti possono eseguire operazioni su una risorsa API Gateway, in base ai tag di cui la risorsa dispone già.
- Quali tag possono essere passati in una richiesta di operazione.
- Se delle chiavi di tag specifiche possono essere utilizzate in una richiesta.

L'utilizzo del controllo degli accessi basato sugli attributi permette un controllo migliore rispetto a quello a livello di API, nonché un controllo più dinamico rispetto al controllo degli accessi basato sulle risorse. Possono essere create policy IAM che consentono o negano un'operazione basata sui tag forniti nella richiesta (tag di richiesta) o tag sulla risorsa su cui è eseguita l'operazione (tag delle risorse). In generale, i tag delle risorse sono per risorse già esistenti. I tag di richiesta sono riservati per la creazione di nuove risorse.

Per la sintassi completa e la semantica delle chiavi di condizione di tag, consulta Controllo degli accessi tramite tag nella Guida per l’utente di IAM.

I seguenti esempi mostrano come specificare le condizioni dei tag nelle policy per gli utenti API Gateway.

Limitare le operazioni in base ai tag delle risorse

La seguente policy di esempio concede agli utenti l'autorizzazione a eseguire tutte le operazioni su tutte le risorse, a condizione che tali risorse non dispongano del tag `Environment` con un valore di `prod`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": "apigateway:*",
      "Resource": "*"
    },
    {
      "Effect": "Deny",
      "Action": [
        "apigateway:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Environment": "prod"
        }
      }
    }
  ]
}
```

Consentire operazioni in base ai tag delle risorse

La seguente policy di esempio consente agli utenti di eseguire tutte le operazioni sulle risorse Gateway API, a condizione che tali risorse dispongano del tag `Environment` con un valore di `Development`. L'istruzione `Deny` impedisce all'utente di modificare il valore del tag `Environment`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ConditionallyAllow",
      "Effect": "Allow",
      "Action": [
        "apigateway:*"
      ],
      "Resource": [
        "arn:aws:apigateway:*::*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Environment": "Development"
        }
      }
    },
```

```
    {
      "Sid": "AllowTagging",
      "Effect": "Allow",
      "Action": [
        "apigateway:*"
      ],
      "Resource": [
        "arn:aws:apigateway:*::/tags/*"
      ]
    },
    {
      "Sid": "DenyChangingTag",
      "Effect": "Deny",
      "Action": [
        "apigateway:*"
      ],
      "Resource": [
        "arn:aws:apigateway:*::/tags/*"
      ],
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:TagKeys": "Environment"
        }
      }
    }
  ]
}
```

Negare operazioni di assegnazione di tag

La seguente policy di esempio consente a un utente di eseguire tutte le operazioni Gateway API, ad eccezione della modifica dei tag.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apigateway:*"
      ],
      "Resource": [
        "*"
```

```
      ],
    },
    {
      "Effect": "Deny",
      "Action": [
        "apigateway:*"
      ],
      "Resource": "arn:aws:apigateway:*::/tags*",
    }
  ]
}
```

Consentire operazioni di assegnazione di tag

La seguente policy di esempio consente a un utente di disporre di tutte le risorse Gateway API e di modificare i tag per tali risorse. Per ottenere i tag per una risorsa, l'utente deve disporre delle autorizzazioni `GET` per quella risorsa. Per aggiornare i tag per una risorsa, l'utente deve disporre delle autorizzazioni `PATCH` per quella risorsa.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apigateway:GET",
        "apigateway:PUT",
        "apigateway:POST",
        "apigateway:DELETE"
      ],
      "Resource": [
        "arn:aws:apigateway:*::/tags/*",
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "apigateway:GET",
        "apigateway:PATCH",
      ],
      "Resource": [
        "arn:aws:apigateway:*::*",
      ]
```

```
    }
  ]
}
```

Riferimenti API

Amazon API Gateway fornisce API per creare e distribuire HTTP e WebSocket API personalizzate. Inoltre, le API API Gateway sono disponibili negli AWS SDK standard.

Se utilizzi una lingua per cui esiste un AWS SDK, potresti preferire utilizzare l'SDK anziché utilizzare direttamente le API REST di API Gateway. Gli SDK semplificano l'autenticazione, si integrano senza problemi nel tuo ambiente di sviluppo e forniscono pratico accesso ai comandi API Gateway.

Ecco dove trovare gli AWS SDK e la documentazione di riferimento dell'API REST di API Gateway:

- Strumenti per Amazon Web Services
- Documentazione di riferimento delle API REST tramite Amazon API Gateway
- Amazon API Gateway WebSocket e riferimento alle API HTTP

Quote e note importanti di Amazon API Gateway

La sezione seguente spiega le quote per Amazon API Gateway. Salvo diversamente specificato, è possibile richiedere di aumentare le quote. Per richiedere un aumento delle quote, è possibile utilizzare Service Quotas o contattare il Centro assistenza AWS.

Quando l'autorizzazione è abilitata su un metodo, la lunghezza massima del metodo ARN (ad esempio`arn:aws:execute-api:{region-id}:{account-id}:{api-id}/{stage-id}/{method}/{resource}/{path}`) è di 1600 byte. I valori dei parametri path (la cui dimensione è determinata in fase di esecuzione) possono far sì che la ARN lunghezza superi il limite. Quando ciò accade, il API client riceve una `414 Request URI too long` risposta.

> **Note**
>
> Ciò limita URI la lunghezza quando vengono utilizzate le politiche delle risorse. Nel caso di private, APIs dove è richiesta una politica delle risorse, ciò limita la URI lunghezza di tutte le informazioni privateAPIs.

Argomenti

- APIQuote a livello di account Gateway, per regione
- HTTPAPIquote
- APIQuote del gateway per la configurazione e l'esecuzione di un WebSocket API
- APIQuote del gateway per la configurazione e l'esecuzione di un REST API
- APIQuote gateway per la creazione, l'implementazione e la gestione di un API
- Note importanti su Amazon API Gateway

APIQuote a livello di account Gateway, per regione

Le seguenti quote si applicano per account e per regione in Amazon API Gateway.

Operazione o risorsa	Quota predefinita	Può essere aumentata
Limita la quota per account, per regione e per HTTP APIs REST APIs callback WebSocket APIs WebSocket APIs	10.000 richieste al secondo (RPS) con una capacità di burst aggiuntiva fornita dall'algoritmo token bucket, utilizzando una capacità massima del bucket di 5.000 richieste. * **Note** La quota burst è determinata dal team di assistenza API Gateway in base alla RPS quota complessiva per l'account nella regione. Non è una quota che un cliente può controllare o per la quale può richiedere modifiche.	Sì
Regionale APIs	600	No
Ottimizzato per Edge APIs	120	No

* Per le seguenti regioni, la quota di accelerazione predefinita è 2500 RPS e la quota burst predefinita è 1250RPS: Africa (Città del Capo), Europa (Milano), Asia Pacifico (Giacarta), Medio Oriente (UAE), Asia Pacifico (Hyderabad), Asia Pacifico (Melbourne), Europa (Spagna), Europa (Zurigo), Israele (Tel Aviv) e Canada occidentale (Calgary), Asia Pacifico (Malesia).

HTTPAPIquote

Le seguenti quote si applicano alla configurazione e all'esecuzione di un HTTP API in Gateway. API

Operazione o risorsa	Quota predefinita	Può essere aumentata
Percorsi per API	300	Sì

Operazione o risorsa	Quota predefinita	Può essere aumentata
Integrazioni per API	300	No
Timeout massimo dell'integrazione	30 secondi	No
Fasi per API	10	Sì
APIMappature a più livelli per dominio	200	No
Tag per fase	50	No
Dimensione totale combinata dei valori della riga di richiesta e dell'inte stazione	10240 byte	No
Dimensione payload	10 MB	No
Domini personali zzati per account per regione	120	Sì
Dimensione del modello di log di accesso	3 KB	No
Inserimento nel CloudWatch log di Amazon Logs	1 MB	No

Operazione o risorsa	Quota predefinita	Può essere aumentata
Autorizzatori per API	10	Sì
Destinatari per autorizzatore	50	No
Ambiti per percorso	10	No
Timeout per l'endpoint JSON Web Key Set	1500 ms	No
Dimensione della risposta dall'endpoint JSON Web Key Set	150000 byte	No
Timeout per l'endpoint di individuazione OpenID Connect	1500 ms	No
Timeout per la risposta della funzione Lambda di autorizzazione	10000 ms	No
VPClink per account per regione	10	Sì
Sottoreti per collegamento VPC	10	Sì

Operazione o risorsa	Quota predefinita	Può essere aumentata
Variabili di fase per ogni fase	100	No
Lunghezza, in caratteri, della chiave in una variabile di fase	64	No
Lunghezza, in caratteri, del valore in una variabile di fase	512	No

APIQuote del gateway per la configurazione e l'esecuzione di un WebSocket API

Le seguenti quote si applicano alla configurazione e all'esecuzione di un in WebSocket API Amazon API Gateway.

Operazione o risorsa	Quota predefinita	Può essere aumentata
Nuove connessioni al secondo per account (in tutto WebSocket APIs) per regione	500	Sì
Connessioni simultanee	Non applicabile *	Non applicabile

Operazione o risorsa	Quota predefinita	Può essere aumentata
AWS Lambda autorizzatori per API	10	Sì
AWS Lambda dimensione del risultato dell'auto rizzatore	8 KB	No
Percorsi per API	300	Sì
Integrazioni per API	300	Sì
Timeout integrazi one	50 millisecondi - 29 secondi per tutti i tipi di integrazione, inclusi Lambda, proxy Lambda, proxy e integrazioni. HTTP HTTP AWS	No
Fasi per API	10	Sì
WebSocket dimensione del telaio	32 KB	No
Dimension i payload del messaggio	128 KB **	No
Dimensione massima del modello di mappatura	300 KB	No
Durata della connessione per WebSocket API	2 ore	No

Operazione o risorsa	Quota predefinita	Può essere aumentata
Timeout per connessione inattiva	10 minuti	No
Lunghezza, in caratteri, URL della forma WebSocket API	4096	No

* API Gateway non impone una quota sulle connessioni simultanee. Il numero massimo di connessioni simultanee è determinato dalla velocità delle nuove connessioni al secondo e dalla durata massima della connessione di due ore. Ad esempio, con la quota predefinita di 500 nuove connessioni al secondo, se i client si connettono alla velocità massima nell'arco di due ore, API Gateway può servire fino a 3.600.000 connessioni simultanee.

** A causa della quota di 32 KB per WebSocket frame, un messaggio di dimensioni superiori a 32 KB deve essere suddiviso in più frame, ciascuno di 32 KB o più piccolo. Si applica ai comandi `@connections`. Se si riceve un messaggio di dimensioni maggiori (o con dimensioni di frame maggiori), la connessione viene chiusa con il codice 1009.

APIQuote del gateway per la configurazione e l'esecuzione di un REST API

Le seguenti quote si applicano alla configurazione e all'esecuzione di un in REST API Amazon API Gateway. Per restapi:import o restapi:put, la dimensione massima del file di definizione è di 6 MB. API

Tutte le API quote per quota possono essere aumentate solo in modo specifico. APIs

Operazione o risorsa	Quota predefinita	Può essere aumentata
I nomi di dominio personalizzati per account per regione	120	Sì
APIMappature a più livelli per dominio	200	No
Lunghezza, in caratteri, del file ottimizzato per i bordi URL API	8192	No
Lunghezza, in caratteri, del simbolo regionale URL API	10240	No
Privato APIs per account per regione	600	No
Lunghezza, in caratteri, della politica delle risorse di API Gateway	8192	Sì
APIchiavi per account per regione	10000	No

Operazione o risorsa	Quota predefinita	Può essere aumentata
Certificati client per account in ogni regione	60	Sì
Autorizzatori per API (AWS Lambda e Amazon Cognito)	10	Sì
Parti della documentazione per API	2000	Sì
Risorse per API	300	Sì
Fasi per API	10	Sì
Variabili di fase per ogni fase	100	No
Lunghezza, in caratteri, della chiave in una variabile di fase	64	No
Lunghezza, in caratteri, del valore in una variabile di fase	512	No
Piani di utilizzo per account in ogni regione	300	Sì
Piani di utilizzo per API chiave	10	Sì

Operazione o risorsa	Quota predefinita	Può essere aumentata
VPClink per account per regione	20	Sì
APImemorizzazione nella cache TTL	300 secondi per impostazione predefinita e configurabile tra 0 e 3600 dal proprietario. API	Non per il limite superiore (3600)
Dimensione della risposta memorizzata nella cache	1048576 byte. la crittografia dei dati della cache può aumentare le dimensioni della voce quando viene memorizzata nella cache.	No
Timeout integrazione	50 millisecondi - 29 secondi per tutti i tipi di integrazione, inclusi Lambda, proxy Lambda, proxy e integrazioni. HTTP HTTP AWS	Sì*
Dimensione totale combinata di tutti i valori di intestazione, inclusi i nomi delle intestazioni, i valori, eventuali terminatori di riga e gli spazi bianchi	10240 byte	No
Dimensione totale combinata di tutti i valori di intestazione per un file privato API	8000 byte	No
Dimensione payload	10 MB	No

Operazione o risorsa	Quota predefinita	Può essere aumentata
Tag per fase	50	No
Numero di iterazioni in un loop `#foreach ... #end` nei modelli di mappatura	1000	No
Dimensione massima del modello di mappatura	300 KB	No
ARNlunghezza di un metodo con autorizzazione	1600 byte	No
Impostazioni di limitazione a livello di metodo per una fase in un piano di utilizzo	20	Sì
Dimensioni del modello per API	400 KB	No
Numero di certificati in un truststore	1.000 certificati con una dimensione totale dell'oggetto fino a 1 MB.	No

* Non è possibile impostare il timeout di integrazione su meno di 50 millisecondi. Puoi aumentare il timeout di integrazione a più di 29 secondi per Regional APIs e PrivateAPIs, ma ciò potrebbe richiedere una riduzione del limite di limitazione a livello di account.

APIQuote gateway per la creazione, l'implementazione e la gestione di un API

Le seguenti quote fisse si applicano alla creazione, alla distribuzione e alla gestione di un API gateway API interno, utilizzando la AWS CLI console Gateway o il API Gateway e il API relativo. REST API SDKs Queste quote non possono essere aumentate.

Operazione	Quota predefinita	Può essere aumentata
CreateApiKey	5 richieste al secondo per account	No
CreateDeployment	1 richiesta ogni 5 secondi per account	No
CreateDocumentationVersion	1 richiesta ogni 20 secondi per account	No
CreateDomainName	1 richiesta ogni 30 secondi per account	No
CreateResource	5 richieste al secondo per account	No
CreateRestApi	Regionale o privato API • 1 richiesta ogni 3 secondi per account Ottimizzato per Edge API • 1 richiesta ogni 30 secondi per account	No
CreateVpcLink(V2)	1 richiesta ogni 15 secondi per account	No

Operazione	Quota predefinita	Può essere aumentata
DeleteApiKey	5 richieste al secondo per account	No
DeleteDomainName	1 richiesta ogni 30 secondi per account	No
DeleteResource	5 richieste al secondo per account	No
DeleteRestApi	1 richiesta ogni 30 secondi per account	No
GetResources	5 richieste ogni 2 secondi per account	No
DeleteVpcLink(V2)	1 richiesta ogni 30 secondi per account	No
ImportDocumentationParts	1 richiesta ogni 30 secondi per account	No
ImportRestApi	Regionale o privato API • 1 richiesta ogni 3 secondi per account Ottimizzato per Edge API • 1 richiesta ogni 30 secondi per account	No
PutRestApi	1 richiesta al secondo per account	No
UpdateAccount	1 richiesta ogni 20 secondi per account	No

Operazione	Quota predefinita	Può essere aumentata
UpdateDomainName	1 richiesta ogni 30 secondi per account	No
UpdateUsagePlan	1 richiesta ogni 20 secondi per account	No
Altre operazioni	Nessuna quota fino alla quota totale dell'account.	No
Operazioni totali	10 richieste al secondo con una quota di ottimizzazione di 40 richieste al secondo.	No

Note importanti su Amazon API Gateway

La sezione seguente descrive in dettaglio le note che potrebbero influire sull'utilizzo di API Gateway.

Argomenti

- Note importanti su Amazon API Gateway per REST APIs HTTPAPIs, e WebSocket APIs
- Note importanti su Amazon API Gateway per HTTP APIs
- Note importanti su Amazon API Gateway per REST e WebSocket APIs
- Note importanti su Amazon API Gateway per WebSocket APIs
- Note importanti su Amazon API Gateway per REST APIs

Note importanti su Amazon API Gateway per REST APIs HTTPAPIs, e WebSocket APIs

- La versione 4A di Signature non è ufficialmente supportata da Amazon API Gateway.

Note importanti su Amazon API Gateway per HTTP APIs

- HTTPAPIstradurrà le `X-Forwarded-*` intestazioni in entrata in un'`Forwarded`intestazione standard e aggiungerà l'IP di uscita, l'host e il protocollo.

Note importanti su Amazon API Gateway per REST e WebSocket APIs

- APIGateway non supporta la condivisione di un nome di dominio personalizzato tra REST e WebSocket APIs.
- I nomi di fasi possono contenere solo caratteri alfanumerici, trattini e caratteri di sottolineatura. La lunghezza massima è 128 caratteri.
- I percorsi di `/ping` e `/sping` sono riservati al controllo dello stato del servizio. Il loro utilizzo per risorse a API livello di root con domini personalizzati non produrrà il risultato previsto.
- APIAttualmente Gateway limita gli eventi di registro a 1024 byte. Gli eventi di registro di dimensioni superiori a 1024 byte, come i corpi di richiesta e risposta, verranno troncati da Gateway prima dell'invio ai log. API CloudWatch
- CloudWatch Attualmente Metrics limita i nomi e i valori delle dimensioni a 255 caratteri validi. XML (Per ulteriori informazioni, consulta la Guida per l'CloudWatch utente.) I valori delle dimensioni sono una funzione dei nomi definiti dall'utente, tra cui API nome, nome dell'etichetta (fase) e nome della risorsa. Quando scegli questi nomi, fai attenzione a non superare i limiti di CloudWatch Metrics.
- La dimensione massima di un modello di mappatura è 300 KB.

Note importanti su Amazon API Gateway per WebSocket APIs

- APIGateway supporta payload di messaggi fino a 128 KB con una dimensione massima dei frame di 32 KB. Se un messaggio supera i 32 KB, occorre suddividerlo in più frame, ciascuno di 32 KB o più piccolo. Se si riceve un messaggio di dimensioni maggiori, la connessione viene chiusa con codice 1009.

Note importanti su Amazon API Gateway per REST APIs

- Il carattere pipe (`|`) in testo semplice non è supportato per nessuna stringa di URL query di richiesta e deve essere codificato in URL -codifica.
- Il carattere punto e virgola (`;`) non è supportato per nessuna stringa di URL query di richiesta e comporta la divisione dei dati.
- RESTAPIsdecodifica i parametri di richiesta URL con codifica prima di passarli alle integrazioni di backend. Per UTF -8 parametri di richiesta, REST APIs decodifica i parametri e poi passali come unicode alle integrazioni di backend.

- Quando usi la console API Gateway per testare un certificatoAPI, potresti ricevere una risposta «errori di endpoint sconosciuti» se al backend viene presentato un certificato autofirmato, se il certificato intermedio non è presente nella catena di certificati o se il backend presenta altre eccezioni irriconoscibili relative ai certificati.
- Per un'`Method`entità API `Resource`or con un'integrazione privata, è necessario eliminarla dopo aver rimosso qualsiasi riferimento codificato di a. `VpcLink` Altrimenti, hai un'integrazione sospesa e ricevi un errore che indica che il VPC link è ancora in uso anche quando l'`Resource`entità or viene eliminata. `Method` Questo comportamento non si applica quando l'integrazione privata fa riferimento a `VpcLink` tramite una variabile di fase.
- I seguenti backend potrebbero non supportare l'autenticazione SSL del client in un modo compatibile con Gateway: API
  - NGINX
  - Heroku
- APIGateway supporta la maggior parte delle specifiche Open API 2.0 e Open API 3.0, con le seguenti eccezioni:
  - I segmenti di percorso possono contenere solo caratteri alfanumerici, trattini, punti, virgole, due punti e parentesi graffe. I parametri di percorso devono essere segmenti di percorso separati. Ad esempio, "risorsa/{nome_parametro_percorso}" è valido, mentre "risorsa{nome_parametro_percorso}" no.
  - I nomi di modelli possono contenere solo caratteri alfanumerici.
  - Per i parametri di input, sono supportati solo i seguenti attributi: `name`, `in`, `required`, `type`, `description`. Altri attributi vengono ignorati.
  - Il tipo `securitySchemes`, se utilizzato, deve essere `apiKey`. Tuttavia, l'autenticazione OAuth 2 e HTTP Basic sono supportate tramite autorizzatori Lambda; la API configurazione Open viene ottenuta tramite le estensioni del fornitore.
  - Il `deprecated` campo non è supportato e viene eliminato in Export. APIs
  - APIl modelli di gateway vengono definiti utilizzando JSONlo schema draft 4, anziché lo JSON schema utilizzato da OpenAPI.
  - Il parametro `discriminator` non è supportato in nessun oggetto dello schema.
  - Il tag `example` non è supportato.
  - `exclusiveMinimum`non è supportato da API Gateway.

- I tag `maxItems` e `minItems` non sono inclusi nella convalida della richiesta semplice. Come soluzione alternativa, aggiorna il modello in seguito all'importazione e prima di procedere con la convalida.
- `oneOf`non è supportato per Open API 2.0 o SDK la generazione.
- Il campo `readOnly` non è supportato.
- `$ref` non può essere utilizzato per fare riferimento ad altri file.
- Le definizioni di risposta del `"500": {"$ref": "#/responses/UnexpectedError"}` modulo non sono supportate nella radice del API documento Open. Come soluzione alternativa, sostituisci i riferimenti con lo schema inline.
- I numeri del tipo `Int32` o `Int64` non sono supportati. Di seguito viene riportato un esempio:

```
"elementId": {
    "description": "Working Element Id",
    "format": "int32",
    "type": "number"
}
```

- Il tipo di formato con numeri decimali (`"format": "decimal"`) non è supportato nelle definizioni dello schema.
- Nelle risposte del metodo, la definizione dello schema deve essere di tipo oggetto e non di tipo di base. Ad esempio, `"schema": { "type": "string"}` non è supportato. Tuttavia, puoi trovare una soluzione alternativa al problema utilizzando il tipo di oggetto seguente:

```
"schema": {
     "$ref": "#/definitions/StringResponse"
            }

 "definitions": {
    "StringResponse": {
      "type": "string"
    }
  }
```

- APIGateway non utilizza la sicurezza di livello root definita nella API specifica Open. Pertanto, la sicurezza deve essere definita a livello di operazione per essere applicata in modo appropriato.
- La parola chiave `default` non è supportata.

- APIGateway applica le seguenti restrizioni e limitazioni quando gestisce i metodi con integrazione o integrazione HTTP Lambda.

- I nomi delle intestazioni e i parametri di query vengono elaborati tenendo presente la distinzione tra lettere maiuscole e minuscole.
- La tabella seguente elenca le intestazioni che potrebbero venire interrotte, rimappate o modificate quando vengono inviate all'endpoint dell'integrazione o reinviate dall'endpoint dell'integrazione: In questa tabella:
  - `Remapped` significa che il nome dell'intestazione è cambiato da `<string>` a `X-Amzn-Remapped-<string>`.

    `Remapped Overwritten` significa che il nome dell'intestazione è cambiato da `<string>` a `X-Amzn-Remapped-<string>` e il valore viene sovrascritto.

Nome intestazione	Richiesta (**`http`**/**`http_proxy`** /**`lambda`**)	Risposta (**`http`**/**`http_proxy`** /**`lambda`**)
`Age`	Transito	Transito
`Accept`	Transito	Interrotta/ Transito/ Transito
`Accept-Charset`	Transito	Transito
`Accept-Encoding`	Transito	Transito
`Authorization`	Transito *	Rimappata
`Connection`	Transito/Transito/Interrotta	Rimappata
`Content-Encoding`	Transito/Interrotta/Transito	Transito

Nome intestazione	Richiesta (**`http`**/**`http_proxy`** /**`lambda`**)	Risposta (**`http`**/**`http_proxy`** /**`lambda`**)
`Content-Length`	Transito (generato in base al corpo)	Transito
`Content-MD5`	Interrotta	Rimappata
`Content-Type`	Transito	Transito
`Date`	Transito	Rimappata Sovrascritta
`Expect`	Interrotta	Interrotta
`Host`	Sovrascritto nell'endpoint di integrazione	Interrotta
`Max-Forwards`	Interrotta	Rimappata
`Pragma`	Transito	Transito
`Proxy-Authenticate`	Interrotta	Interrotta
`Range`	Transito	Transito
`Referer`	Transito	Transito
`Server`	Interrotta	Rimappata Sovrascritta
`TE`	Interrotta	Interrotta

Nome intestazione	Richiesta (**`http`**/**`http_proxy`** /**`lambda`**)	Risposta (**`http`**/**`http_proxy`** /**`lambda`**)
`Transfer-Encoding`	Interrotta/Interrotta/Eccezione	Interrotta
`Trailer`	Interrotta	Interrotta
`Upgrade`	Interrotta	Interrotta
`User-Agent`	Transito	Rimappata
`Via`	Interrotta/Interrotta/Transito	Transito/Interrotta/Interrotta
`Warn`	Transito	Transito
`WWW-Authenticate`	Interrotta	Rimappata

* L'intestazione `Authorization` viene eliminata se contiene una firma Signature Version 4 o viene usata un'autorizzazione `AWS_IAM`.

- L'Android SDK di un API generato da API Gateway utilizza la `java.net.HttpURLConnection` classe. Questa classe genererà un'eccezione non gestita, sui dispositivi con Android 4.4 e versioni precedenti, per la risposta 401 derivante dalla rimappatura dell'intestazione `WWW-Authenticate` su `X-Amzn-Remapped-WWW-Authenticate`.
- A differenza di Java, Android e iOS SDKs di an API generati da GatewayAPI, quelli API generati da API Gateway non supportano nuovi tentativi per errori di livello 500. JavaScript SDK
- La chiamata di test di un metodo utilizza il tipo di contenuto predefinito di `application/json` e ignora le specifiche di altri tipi di contenuto.

- Quando si inviano richieste a un API tramite l'`X-HTTP-Method-Overrideintestazione`, Gateway sovrascrive il metodo. API Per passare l'intestazione al back-end, è necessario aggiungere l'intestazione all'integrazione richiesta.
- Quando una richiesta contiene più tipi di file multimediali nell'`Acceptintestazione`, API Gateway rispetta solo il primo tipo di supporto. `Accept` Nel caso in cui non sia possibile controllare l'ordine dei tipi di `Accept` file multimediali e il tipo di supporto del contenuto binario non sia il primo nell'elenco, è possibile aggiungere il primo tipo di `Accept` file multimediale nell'`binaryMediaTypeselencoAPI`, API Gateway restituirà il contenuto come binario. Ad esempio, per inviare un JPEG file utilizzando un `<img>` elemento in un browser, il browser potrebbe inviare `Accept:image/webp,image/*,*/*;q=0.8` una richiesta. Aggiungendo `image/webp` all'`binaryMediaTypeselenco`, l'endpoint riceverà il JPEG file come binario.
- La personalizzazione della risposta del gateway predefinito per `413 REQUEST_TOO_LARGE` non è attualmente supportata.
- APIGateway include un'`Content-Typeintestazione` per tutte le risposte di integrazione. Per impostazione predefinita, il tipo di contenuto è `application/json`.

Cronologia dei documenti

La tabella seguente descrive le modifiche importanti alla documentazione dall'ultima versione di Amazon API Gateway. Per ricevere notifiche sugli aggiornamenti di questa documentazione, puoi iscriverti a un RSS feed scegliendo il RSS pulsante nel pannello del menu in alto.

- Ultimo aggiornamento della documentazione: 15 febbraio 2024

Modifica	Descrizione	Data
È stato aggiunto il supporto per 1.3 TLS	APIGateway ora supporta TLS 1.3 su Regional REST APIs HTTPAPIs, e WebSocket APIs. Per ulteriori informazi oni, consulta Scelta di una politica di sicurezza per il dominio personalizzato in API Gateway.	15 febbraio 2024
RESTAPIe aggiornamenti WebSocket API della console	Informazioni aggiornate sulla console per REST APIs e WebSocket APIs.	10 dicembre 2023
Aggiornamento della documentazione	Informazioni concettuali aggiornate e creazione di nuovi tutorial per le trasforma zioni dei dati e argomenti di convalida delle richieste per Gateway. API REST APIs Per ulteriori informazioni, consulta Utilizzare la convalida delle richieste in API Gateway e Configurazione delle trasforma zioni dei dati per. REST API	22 giugno 2023
Configurare il DNS failover per un gateway multiregionale API	È stato aggiunto il supporto per utilizzare i controlli di	31 ottobre 2022

	integrità di Amazon Route 53 per controllare il DNS failover da un API gateway REST API in una regione primaria Regione AWS a uno in una regione secondaria. Per ulteriori informazioni, consulta Configurare controlli sanitari personalizzati per il DNS failover.	
Aggiornamento della documentazione	Riepiloghi aggiornati delle funzionalità principali per REST API e. HTTP API APIs Per ulteriori informazioni, consulta Scelta tra REST API e HTTP API APIs.	31 maggio 2022
Aggiornamento della policy gestita	Aggiunto supporto `acm:GetCertificate` per la policy `AWSServiceRoleForAPIGateway`. Per ulteriori informazioni, vedere Utilizzo dei ruoli collegati ai servizi per API Gateway.	12 luglio 2021
Mappatura dei parametri per HTTP APIs	È stato aggiunto il supporto per la mappatura dei parametri per. HTTP APIs Per ulteriori informazioni, consulta Trasformazione di API richieste e risposte.	7 gennaio 2021

Disabilita l'endpoint predefinito per a REST API	È stato aggiunto il supporto per la disabilitazione dell'endpoint predefinito per. REST APIs Per ulteriori informazioni, vedere Disabilitazione dell'endpoint predefinito per a. REST API	29 ottobre 2020
Autenticazione reciproca TLS	È stato aggiunto il supporto per TLS l'autenticazione reciproca per REST APIs e HTTPAPIs. Per ulteriori informazioni, vedere Configurazione dell'TLSautenticazione reciproca per a REST API e Configurazione dell'TLSautenticazione reciproca per un. HTTP API	17 settembre 2020
HTTPAPI AWS Lambda autorizzatori	È stato aggiunto il supporto per gli AWS Lambda autorizzatori per. HTTP APIs Per ulteriori informazioni, consulta Lavorare con gli AWS Lambda autorizzatori per. HTTP APIs	9 settembre 2020
HTTPAPI AWS integrazioni di servizi	È stato aggiunto il supporto per le integrazioni AWS di servizi per. HTTP APIs Per ulteriori informazioni, consulta Lavorare con le integrazioni AWS di servizi per. HTTP APIs	20 agosto 2020

HTTPAPIdomini personalizzati wildcard	È stato aggiunto il supporto per i nomi di dominio personalizzati wildcard per. HTTP APIs Per ulteriori informazioni, consulta Nomi di dominio personalizzati con caratteri jolly.	10 agosto 2020
Miglioramenti del portale per sviluppatori serverless	È stata aggiunta la gestione degli utenti al pannello di amministrazione e il supporto per l'esportazione delle definizioni. API Per ulteriori informazioni, consulta Usa il portale per sviluppatori serverless per catalogare il tuo API Gateway. APIs	25 giugno 2020
WebSocket API`Sec-WebSocket-Protocol` supporto	Aggiunta del supporto per il campo `Sec-WebSocket-Protocol` Per ulteriori informazioni, vedi Configurazione di una route $connect che richiede un WebSocket sottoprotocollo.	16 giugno 2020
HTTPAPIesportare	È stato aggiunto il supporto per l'esportazione delle definizioni Open API 3.0 di HTTPAPIs. Per ulteriori informazioni, vedere Esportazione di un file HTTP API da API Gateway.	20 aprile 2020

Documentazione sulla sicurezza	Aggiunta documentazione sulla sicurezza Per ulteriori informazioni, consulta la sezione Sicurezza in Amazon API Gateway.	31 marzo 2020
Documentazione riorganizzata.	Riorganizzata la guida per sviluppatori.	12 marzo 2020
HTTPAPIdisponibilità generale	HTTPAPIsRilasciato in disponibilità generale. Per ulteriori informazioni, consulta Lavorare con HTTP APIs.	12 marzo 2020
HTTPAPIregistrazione	È stato aggiunto il supporto per i HTTP API log `$context.integrationErrorMessage` in. Per ulteriori informazioni, consulta HTTPAPILogging Variables.	26 febbraio 2020
AWS variabili per Open import API	È stato aggiunto il supporto per AWS le variabili nelle API definizioni Open. Per ulteriori informazioni, vedete AWS Variabili per l'APIimportazione aperta.	17 febbraio 2020
HTTP APIs	Rilasciato HTTP APIs in versione beta. Per ulteriori informazioni, vedere HTTPAPIs.	4 dicembre 2019

Nomi di dominio personalizzati con caratteri jolly	Aggiunto il supporto per i nomi di dominio personalizzati con caratteri jolly. Per ulteriori informazioni, consulta Nomi di dominio personalizzati con caratteri jolly.	21 ottobre 2019
Registrazione di Amazon Data Firehose	È stato aggiunto il supporto per Amazon Data Firehose come destinazione per i dati di registrazione degli accessi. Per ulteriori informazioni, consulta Utilizzo di Amazon Data Firehose come destinazione per la registrazione degli accessi al API gateway.	15 ottobre 2019
Alias Route53 per invocare private APIs	È stato aggiunto il supporto per record di alias Route53 aggiuntivi per l'invocazione di private. DNS APIs Per ulteriori informazioni, vedere Accesso privato API tramite alias Route53.	18 settembre 2019
Controllo degli accessi basato su tag per WebSocket APIs	È stato aggiunto il supporto per il controllo degli accessi basato su tag per. WebSocket APIs Per ulteriori informazioni, consulta APIGateway Resources That Can Be Tagged.	27 giugno 2019

TLSselezione della versione per domini personalizzati	È stato aggiunto il supporto per la selezione delle versioni di Transport Layer Security (TLS) APIs che vengono distribuite su domini personali zzati. Vedi la nota in Scegli una TLS versione minima per un dominio personalizzato in API Gateway.	20 giugno 2019
VPCpolitiche degli endpoint per uso privato APIs	È stato aggiunto il supporto per migliorare la sicurezza degli endpoint privati APIs collegando le policy degli endpoint agli endpoint di interfaccia. VPC Per ulteriori informazioni, consulta Use VPC Endpoint Policies for Private in Gateway. APIs API	4 giugno 2019
Documentazione aggiornata	Guida introduttiva riscritta ad Amazon API Gateway. Tutorial spostati in Amazon API Gateway Tutorials.	29 maggio 2019
Controllo degli accessi basato su tag per REST APIs	È stato aggiunto il supporto per il controllo degli accessi basato su tag per. REST APIs Per ulteriori informazi oni, vedere Utilizzo di tag con IAM politiche per controllare l'accesso alle risorse del API gateway.	23 maggio 2019

Documentazione aggiornata	6 argomenti riscritti: Cos'è Amazon API Gateway? , Tutorial: Crea un modello API con HTTP Proxy Integration, Tutorial: Crea un Calc REST API con tre integrazioni non proxy, APIGateway Mapping Template e Access Logging Variable Reference , Usa Gateway API Lambda Authorizers e Enable for an Gateway Resource. CORS API REST API	5 aprile 2019
Miglioramenti del portale per sviluppatori serverless	Aggiunto il pannello di amministrazione e altri miglioramenti per semplificare la pubblicazione APIs nel portale per sviluppatori Amazon API Gateway. Per ulteriori informazioni, consulta Usa un portale per sviluppatori per catalogare i tuoi APIs.	28 marzo 2019
Support per AWS Config	È stato aggiunto il supporto per AWS Config. Per ulteriori informazioni, vedere Monitoring API Gateway API Configuration con AWS Config.	20 marzo 2019
Support per AWS CloudFormation	Aggiunto API Gateway V2 API al riferimento del AWS CloudFormation modello. Per ulteriori informazioni, consulta Amazon API Gateway V2 Resource Types Reference.	7 febbraio 2019

Support per WebSocket APIs	È stato aggiunto il supporto per WebSocket APIs. Per ulteriori informazioni, consulta Creazione di un API gateway WebSocket API in Amazon.	18 dicembre 2018
Portale per sviluppatori serverless disponibile tramite AWS Serverless Application Repository	L'applicazione serverless del portale per sviluppatori Amazon API Gateway è ora disponibile presso AWS Serverless Application Repository(in aggiunta a GitHub). Per ulteriori informazioni, consulta Usa un portale per sviluppatori per catalogare il tuo API gateway APIs.	16 novembre 2018
Support per AWS WAF	Aggiunto il supporto per AWS WAF (Web Application Firewall). Per ulteriori informazioni, vedere Controllare l'accesso a un API utilizzo AWS WAF.	5 Novembre 2018
Portale per gli sviluppatori serverless	Amazon API Gateway ora fornisce un portale per sviluppatori completamente personalizzabile come applicazione serverless che puoi distribuire per pubblicare il tuo API Gateway. APIs Per ulteriori informazioni, consulta Usa un portale per sviluppatori per catalogare il tuo API gateway. APIs	29 ottobre 2018

Supporto per le intestazioni multi-valore e i parametri di stringa di query	Amazon API Gateway ora supporta più header e parametri di stringhe di query con lo stesso nome. Per ulteriori informazioni, consulta la sezione relativa al supporto per le intestazioni multi-valore e i parametri di stringa di query.	4 ottobre 2018
Supporto aperto API	Amazon API Gateway ora supporta Open API 3.0 e Open API (Swagger) 2.0.	27 settembre 2018
Documentazione aggiornata	È stato aggiunto un nuovo argomento: In che modo le politiche delle risorse di Amazon API Gateway influiscono sul flusso di lavoro di autorizzazione.	27 settembre 2018
AWS X-Ray Integrazione attiva	Ora puoi utilizzarla AWS X-Ray per tracciare e analizzare le latenze nelle richieste degli utenti man mano che arrivano APIs ai servizi sottostanti. Per ulteriori informazioni, consulta Trace API Gateway API Execution with AWS X-Ray.	6 settembre 2018

Miglioramenti caching	Solo GET i metodi avranno la memorizzazione nella cache abilitata per impostazione predefinita quando abiliti la memorizzazione nella cache per uno API stage. Questo aiuta a garantire la sicurezza del tuo. API Puoi abilitare il caching per altri metodi ignorando le impostazioni del metodo. Per ulteriori informazioni, consulta Abilita la API memorizzazione nella cache per migliorare la reattività.	20 agosto 2018
Restrizioni dei servizi aggiornate	Sono stati modificati diversi limiti: aumento del numero di APIs account. Limiti API di velocità aumentati per APIs Create/Import/Deploy. Alcuni tassi sono stati corretti da al minuto in al secondo. Per ulteriori informazioni, consulta Limiti.	13 luglio 2018
Sovrascrivere API i parametri e le intestazioni di richiesta e risposta	Aggiunto il supporto per sovrascrivere intestazioni di richiesta, stringhe di query e percorsi, nonché intestazioni di risposta e codici di stato. Per ulteriori informazioni, consulta Utilizzare un modello di mappatura per sovrascrivere i parametri e le intestazioni API di richiesta e risposta di un utente.	12 luglio 2018

Throttling a livello di metodo per piani di utilizzo	È stato aggiunto il supporto per l'impostazione dei limiti di limitazione predefiniti per metodo, nonché per l'imposta zione dei limiti di limitazio ne per i singoli metodi nelle impostazioni del piano di utilizzo. API Queste impostazi oni sono in aggiunta ai limiti di throttling a livello di account e ai limiti di throttling a livello di metodo esistenti che puoi definire nelle impostazi oni della fase. Per ulteriori informazioni, consulta Throttle Requests for Better Throughpu t. API	11 luglio 2018
APILe notifiche di aggiornam ento della Gateway Developer Guide sono ora disponibili tramite RSS	La HTML versione della API Gateway Developer Guide ora supporta un RSS feed di aggiornamenti documentati in questa pagina della cronologi a dei documenti. Il RSS feed include gli aggiornamenti effettuati il 27 giugno 2018 e versioni successive. Gli aggiornamenti annunciati in precedenza sono ancora disponibili in questa pagina. Usa il RSS pulsante nel pannello del menu in alto per iscriverti al feed.	27 giugno 2018

Aggiornamenti precedenti

La tabella seguente descrive le modifiche importanti apportate in ogni versione della APIGateway Developer Guide prima del 27 giugno 2018.

Modifica	Descrizione	Data della modifica
Privato APIs	È stato aggiunto il supporto per private APIs, che è possibile esporre tramite gli VPCendpoint dell'interfaccia. Il traffico verso la tua rete privata APIs non esce dalla rete Amazon; è isolato dalla rete Internet pubblica.	14 giugno 2018
Provider di autorizzazioni e integrazioni Lambda tra più account e provider di autorizza zioni per il pool di utenti di Amazon Cognito multiacco unt	Usa una AWS Lambda funzione di un AWS account diverso come funzione di autorizzazione Lambda o come backend di API integrazione. Oppure utilizzare un pool di utenti di Amazon Cognito come un provider di autorizza zioni. L'altro account può trovarsi in qualsiasi regione in cui Amazon API Gateway è disponibile. Per ulteriori informazi oni, consulta the section called “Configurazione di un'autori zzazione Lambda tra account”, the section called “Tutorial : Creazione di un'RESTAPIintegrazione proxy Lambda con più account” e the section called “Configurazione dell'auto rizzazione Amazon Cognito tra account per un'API REST”.	2 Aprile 2018
Politiche relative alle risorse per APIs	Utilizza le policy relative alle risorse del API gateway per consentire agli utenti di un altro AWS account di accedere in modo sicuro al tuo account API o per consentire che vengano richiamate solo da intervalli o CIDR blocchi di indirizzi IP di origine specificati. API Per ulteriori informazi oni, consulta the section called “Utilizza le politiche delle risorse del API Gateway”.	2 aprile 2018
Etichettatura per le risorse Gateway API	Aggiungi un tag a una API fase con un massimo di 50 tag per l'allocazione dei costi delle API richieste e la memorizzazione nella cache in Gateway. API Per ulteriori	19 dicembre 2017

Modifica	Descrizione	Data della modifica
	informazioni, consulta the section called “Configurazione dei tag”.	
Compressione e decompressione dei payload	Abilita le chiamate API con payload compressi utilizzan do una delle codifiche di contenuto supportate. I payload compressi sono soggetti a mappatura se è specificato il modello di mappatura del corpo. Per ulteriori informazioni, consulta the section called “Codifica del contenuto”.	19 dicembre 2017
APIchiave proveniente da un autorizzatore personalizzato	Restituisci una API chiave da un autorizzatore personali zzato a API Gateway per applicare un piano di utilizzo per API i metodi che richiedono la chiave. Per ulteriori informazioni, consulta the section called “Scegli una fonte API chiave in API Gateway”.	19 dicembre 2017
Autorizzazione con OAuth 2 ambiti	Abilita l'autorizzazione all'invocazione del metodo utilizzan do OAuth 2 ambiti con l'autorizzatore. `COGNITO_U SER_POOLS` Per ulteriori informazioni, consulta the section called “Utilizza un pool di utenti di Amazon Cognito come autorizzazione per un'API REST”.	14 dicembre 2017
Integrazione privata e collegamento VPC	Crea un'integrazione privata API con API Gateway per fornire ai clienti l'accesso HTTP a/alle HTTPS risorse in un Amazon VPC dall'esterno o VPC tramite una VpcLinkrisorsa. Per ulteriori informazioni, consulta the section called “Tutorial: crea un'API REST con un'integr azione privata” e the section called “Integrazione privata”.	30 Novembre 2017

Modifica	Descrizione	Data della modifica
Implementa un Canary per i test API	Aggiungi una versione Canary a una API distribuzione esistente per testare una versione più recente di, API mantenendo la versione corrente operativa sullo stesso stadio. Puoi impostare una percentuale di traffico di stage per la versione Canary e abilitare l'esecuzione e l'accesso specifici di Canary registrati in registri di Logs separati. CloudWatch Per ulteriori informazioni, consulta the section called “Configurazione della distribuzione di una release Canary”.	28 Novembre 2017
Logging degli accessi	Registra l'accesso del client al tuo account API con i dati derivati dalle variabili $context in un formato a tua scelta. Per ulteriori informazioni, consulta the section called “CloudWatch registri”.	21 Novembre 2017
Ruby di un SDK API	Genera un Ruby SDK per te API e usalo per richiamare i tuoi metodi. API Per ulteriori informazioni, consulta the section called “Genera il Ruby SDK di un in Gateway API API” e the section called “Usa un Ruby SDK generato da API Gateway per un REST API”.	20 novembre 2017
Endpoint regionale API	Specificare un API endpoint regionale per crearne uno API per client non mobili. Un client non mobile, ad esempio un'EC2istanza, viene eseguito nella stessa AWS regione in cui viene distribuito. API Come con un sistema ottimizza to per l'edgeAPI, puoi creare un nome di dominio personali zzato per una regione. API Per ulteriori informazioni, consulta the section called “Tipi di endpoint API Gateway” e the section called “Configura un nome di dominio personali zzato regionale”.	2 Novembre 2017

Modifica	Descrizione	Data della modifica
Autorizzazioni ad hoc per le richieste	Utilizza l'autorizzatore di richiesta personalizzato per fornire informazioni di autenticazione dell'utente nei parametri di richiesta per autorizzare le chiamate ai metodi. API I parametri di richiesta includono intestazioni e parametri della stringa di query, nonché variabili di fase e di contesto. Per ulteriori informazioni, consulta Usa gli API autorizzatori Gateway Lambda.	15 settembre 2017
Personalizzazione delle risposte del gateway	Personalizza le risposte del API gateway generate dal gateway alle API richieste che non sono riuscite a raggiungere il backend di integrazione. Un messaggio gateway personalizzato può fornire al chiamante messaggi di errore personalizzati API specifici, inclusa la restituzione delle CORS intestazioni necessarie, oppure può trasforma re i dati di risposta del gateway in un formato di scambio esterno. Per ulteriori informazioni, consulta Configurazione delle risposte del gateway per la personalizzazione delle risposte agli errori.	6 giugno 2017
Mappatura di proprietà di errore personalizzate di Lambda a intestazi oni di risposte dei metodi	Mappa le singole proprietà di errore personalizzate restituite da Lambda ai parametri dell'intestazione della risposta del metodo utilizzando il `integration.respon se.body` parametro, affidando a API Gateway la deserializzazione dell'oggetto di errore personalizzato con stringhe in fase di esecuzione. Per ulteriori informazioni, consulta Gestione degli errori Lambda personalizzati in API Gateway.	6 giugno 2017
Aumento dei limiti di throttling	Come aumentare il limite del tasso di richieste con stato fisso a livello di account a 10.000 richieste al secondo (rps) e il limite di aumento delle prestazioni a 5000 richieste simultanee. Per ulteriori informazioni, consulta Limita le richieste alle tue API REST per una migliore velocità di trasmissione in API Gateway.	6 giugno 2017

Modifica	Descrizione	Data della modifica
Convalida delle richieste di metodi	Configura i validatori di richieste di base a API livello o a livello di metodo in modo che Gateway possa convalida re le richieste in arrivo. API APIGateway verifica che i parametri richiesti siano impostati e non vuoti e verifica che il formato dei payload applicabili sia conforme al modello configurato. Per ulteriori informazioni, consulta Richiedi la convalida per REST APIs in API Gateway.	11 aprile 2017
Integrazione con ACM	Usa ACM Certificates per i tuoi nomi API di dominio personalizzati. Puoi creare un certificato in AWS Certifica te Manager o importare un certificato in PEM formato esistente in. ACM Quindi fai riferimento ai certificati ARN quando imposti un nome di dominio personalizzato per il tuo. APIs Per ulteriori informazioni, consulta Nome di dominio personalizzato per REST APIs in API Gateway.	9 marzo 2017
Generazione e chiamata di un file Java SDK di un API	Consenti a API Gateway di generare Java SDK per te API e usalo SDK per chiamarlo API nel tuo client Java. Per ulteriori informazioni, consulta Utilizzo di un SDK Java generato da API Gateway per un'API REST.	13 gennaio 2017
Integrazione con Marketplace AWS	Vendi il tuo API in un piano di utilizzo come prodotto SaaS tramite. Marketplace AWS Marketplace AWS Usalo per estendere la portata del tuoAPI. Affidati alla fatturazione ai clienti Marketplace AWS per tuo conto. Lascia che API Gateway gestisca l'autorizzazione degli utenti e la misurazione dell'utilizzo. Per ulteriori informazioni, consulta Vendi le tue API come SaaS.	1 dicembre 2016

Modifica	Descrizione	Data della modifica
Attivazione del Documentation Support per API	Aggiungi la documentazione per API le entità nelle DocumentationPartrisorse di API Gateway. Associa un'istantanea delle `DocumentationPart` istanze della raccolta a uno API stage per creare un. DocumentationVersion Pubblicate API la documentazione esportando una versione della documentazione in un file esterno, ad esempio un file Swagger. Per ulteriori informazioni, consulta Documentazione per REST APIs in API Gateway.	1 dicembre 2016
Autorizzazioni ad hoc aggiornate	Una funzione Lambda delle autorizzazioni ad hoc restituisce ora l'identificatore dell'entità principale dell'intermediario. La funzione può anche restituire altre informazioni come coppie chiave-valore della mappa e una politica. `context` IAM Per ulteriori informazioni, consulta Output da un autorizzatore API Gateway Lambda.	1 dicembre 2016
Supporto di payload binari	Imposta binaryMediaTypessu di te API per supportare i payload binari di una richiesta o di una risposta. Imposta la `contentHandling` proprietà su un'integrazione o IntegrationResponsespecifica se gestire un payload binario come blob binario nativo, come stringa codificata in Base64 o come passthrough senza modifiche. Per ulteriori informazioni, consulta Tipi di supporti binari per REST APIs in API Gateway.	17 Novembre 2016
Abilitazione di un'integrazione proxy con un backend HTTP o Lambda tramite una risorsa proxy di un. API	Come creare una risorsa proxy con un parametro di percorso greedy in formato `{proxy+}` e il metodo `ANY` catch-all. La risorsa proxy è integrata con un backend HTTP o Lambda utilizzando rispettivamente l'integrazione del proxy o HTTP Lambda. Per ulteriori informazioni, consulta Configurazione di un'integrazione proxy mediante una risorsa proxy.	20 settembre 2016

Modifica	Descrizione	Data della modifica
Estendete l'offerta di prodotti selezionati APIs in API Gateway per i vostri clienti fornendo uno o più piani di utilizzo.	Crea un piano di utilizzo in API Gateway per consentire ai API clienti selezionati di accedere a API fasi specifiche a tariffe e quote di richiesta concordate. Per ulteriori informazioni, consulta Piani di utilizzo e API chiavi per REST APIs in API Gateway .	11 agosto 2016
Abilitazione dell'autorizzazione a livello di metodo con un pool di utenti in Amazon Cognito	Come creare un pool di utenti in Amazon Cognito e usarlo come provider di identità personale. Puoi configurare il pool di utenti come autorizzazione a livello di metodo per concedere l'accesso agli utenti registrati con il pool di utenti. Per ulteriori informazioni, consulta Controlla l'accesso alle API REST utilizzando i pool di utenti di Amazon Cognito come autorizzatore.	28 luglio 2016
Abilitazione CloudWatch delle metriche e delle dimensioni di Amazon nel `AWS/ApiGateway` namespace.	Le metriche API Gateway sono ora standardizzate nel namespace di. CloudWatch `AWS/ApiGateway` Puoi visualizzarli sia nella console API Gateway che nella CloudWatch console Amazon. Per ulteriori informazioni, consulta Dimensioni e parametri di Amazon API Gateway.	28 luglio 2016
Abilitazione della rotazione dei certificati per un nome di dominio personalizzato	La rotazione dei certificati permette di caricare e rinnovare un certificato in scadenza per un nome di dominio personalizzato. Per ulteriori informazioni, consulta Rotazione di un certificato importato in ACM.	27 Aprile 2016

Modifica	Descrizione	Data della modifica
Documentazione delle modifiche per la console Amazon API Gateway aggiornata.	Scopri come creare e configurare una API console API Gateway aggiornata. Per ulteriori informazioni, consulta Tutorial: creazione di un'API REST mediante l'importazione di un esempio e Tutorial: creare un file REST API con un'HTTPintegrazione non proxy.	5 Aprile 2016
Attivazione della API funzionalità di importazione per creare una nuova API definizione o aggiornarne una esistente API da fonti esterne.	Con le API funzionalità di importazione, è possibile crearne una nuova API o aggiornarne una esistente caricando una API definizione esterna espressa in Swagger 2.0 con le API estensioni Gateway. Per ulteriori informazioni sull'importazioneAPI, vedere. Sviluppa REST APIs utilizzando Open API in API Gateway	5 aprile 2016
Esposizione della `$input.body` variabile per accedere al payload non elaborato come stringa e della `$util.parseJson()` funzione per trasformare una JSON stringa in un JSON oggetto in un modello di mappatura.	Per ulteriori informazioni su `$input.body` e `$util.parseJson()` , consulta APIModello di mappatura del gateway e riferimento alla variabile di registrazione degli accessi.	5 Aprile 2016

Modifica	Descrizione	Data della modifica
Abilitazione delle richieste client con invalidamento della cache a livello di metodo e miglioramento della gestione del throttling delle richieste.	Svuota la cache a API livello di fase e invalida l'immissione della cache individuale. Per ulteriori informazioni, consulta Svuota la cache dello API stage in Gateway API e Invalidare una voce della cache del Gateway API. Migliora l'esperienza della console per la gestione della limitazione delle richieste. API Per ulteriori informazioni, consulta Limita le richieste alle tue API REST per una migliore velocità di trasmissione in API Gateway.	25 marzo 2016
Abilitazione e chiamata a API Gateway API utilizzando l'autorizzazione personalizzata	Crea e configura una AWS Lambda funzione per implementare l'autorizzazione personalizzata. La funzione restituisce un documento di IAM policy che concede le autorizzazioni Consenti o Rifiuta alle richieste dei client di un Gateway. API API Per ulteriori informazioni, consulta Usa gli API autorizzatori Gateway Lambda.	11 febbraio 2016
Importazione ed esportazione di API Gateway API utilizzando un file di definizione Swagger e delle estensioni	Crea e aggiorna il tuo API Gateway API utilizzando le specifiche Swagger con le estensioni Gateway. API Importa le definizioni Swagger utilizzando Gateway Importer. API Esporta un API Gateway API in un file di definizione Swagger utilizzando la console Gateway o API Gateway Export. API API Per ulteriori informazioni, consulta Sviluppa REST APIs utilizzando Open API in API Gateway e Esporta un file REST API da API Gateway.	18 dicembre 2015
Mappatura del corpo della richiesta o della risposta o dei JSON campi del corpo in base ai parametri di richiesta o risposta.	Mappa il corpo della richiesta del metodo o JSON i relativi campi nel percorso, nella stringa di query o nelle intestazioni della richiesta di integrazione. Mappa il corpo della risposta di integrazione o JSON i relativi campi nelle intestazioni della risposta alla richiesta. Per ulteriori informazioni, consulta Riferimento alla mappatura dei dati di API richiesta e risposta di Amazon API Gateway.	18 dicembre 2015

Modifica	Descrizione	Data della modifica
Utilizzo delle variabili di fase in Amazon API Gateway	Scopri come associare gli attributi di configurazione a una fase di distribuzione di un API Amazon API Gateway. Per ulteriori informazioni, consulta Usa le variabili di fase per un REST API in API Gateway.	5 Novembre 2015
Procedura: abilitare CORS un metodo	Ora è più semplice abilitare la condivisione di risorse tra origini diverse (CORS) per i metodi in Amazon API Gateway. Per ulteriori informazioni, consulta CORSper REST APIs in API Gateway.	3 Novembre 2015
Procedura: utilizzar e l'autenticazione lato SSL client	Usa Amazon API Gateway per generare SSL certificati da utilizzare per autenticare le chiamate al HTTP backend. Per ulteriori informazioni, consulta Genera e configura un SSL certificato per l'autenticazione backend in API Gateway.	22 settembre 2015
Integrazione fittizia di metodi	Scopri come simulare l'integrazione di un con API Amazon Gateway. API Questa funzionalità consente agli sviluppat ori di generare API risposte direttamente da API Gateway senza la necessità di un backend di integrazione finale.	1 settembre 2015
Supporto Amazon Cognito Identity	Amazon API Gateway ha ampliato l'ambito della `$context` variabile in modo che ora restituisca informazioni su Amazon Cognito Identity quando le richieste vengono firmate con le credenziali di Amazon Cognito. Inoltre, abbiamo aggiunto una `$util` variabile per l'escape dei caratteri e la codifica JavaScript e le stringhe. URLs Per ulteriori informazioni, consulta APIModello di mappatura del gateway e riferimento alla variabile di registrazione degli accessi.	28 agosto 2015

Modifica	Descrizione	Data della modifica
Integrazione di Swagger	Utilizza lo strumento di importazione Swagger GitHub per importare le definizioni Swagger API in Amazon Gateway. API Scopri di più sulla creazione e APIEstensioni aperte per API Gateway la distribuzione e sui metodi di utilizzo dello strumento di APIs importazione. Con lo strumento di importazione Swagger puoi anche aggiornare quelli esistenti. APIs	21 luglio 2015
Informazioni di riferimento sui modelli di mappatura	Informazioni sul parametro `$input` e le relative funzioni in APIModello di mappatura del gateway e riferimento alla variabile di registrazione degli accessi.	18 luglio 2015
Versione pubblica iniziale	Questa è la versione pubblica iniziale della APIGateway Developer Guide.	9 luglio 2015

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.