Panduan Developer, Versi 1

AWS IoT Greengrass

AWS IoT Greengrass: Panduan Developer, Versi 1

Table of Contents

..... xxi
Apakah AWS IoT Greengrass itu? 1
AWS IoT Greengrass Perangkat lunak Core 3
AWS IoT Greengrass Versi perangkat lunak core 4
AWS IoT Greengrass Grup 14
Perangkat di AWS IoT Greengrass 16
SDK 19
Platform dan persyaratan yang didukung 20
AWS IoT Greengrass unduh 33
AWS IoT Greengrass Perangkat lunak Core 34
AWS IoT Greengrass perangkat lunak snap 42
AWS IoT Greengrass Perangkat lunak Docker 42
AWS IoT Greengrass Core SDK 45
Waktu aktif dan perpustakaan machine learning yang didukung 45
AWS IoT Greengrass Perangkat lunak ML SDK 47
Kami ingin mendengar pendapat Anda 47
Instal AWS IoT Greengrass perangkat lunak Core 47
Mengunduh dan mengekstraksi file tar.gz 48
Jalankan skrip pengaturan perangkat Greengrass 48
Instal dari repositori APT 48
Jalankan AWS IoT Greengrass dalam kontainer Docker 50
Jalankan AWS IoT Greengrass dalam snap 51
Arsip instalasi perangkat lunak Core 63
Mengonfigurasi AWS IoT Greengrass core 64
AWS IoT Greengrass file konfigurasi core 65
Titik akhir Anda harus sesuai dengan jenis sertifikat 129
Connect pada port 443 atau melalui proksi jaringan 130
Mengonfigurasi direktori tulis 141
Konfigurasikan pengaturan MQTT 145
Aktifkan deteksi IP otomatis 163
Mulai Greengrass pada boot sistem 166
Lihat juga 167
AWS IoT Greengrass V1kebijakan pemeliharaan 169
AWS IoT Greengrassskema pembuatan versi 169

Fase siklus hidup untuk perangkat lunak Core AWS IoT Greengrass 170
Kebijakan pemeliharaan untuk perangkat lunak AWS IoT Greengrass Inti 170
Jadwal fase pemeliharaan 171
Jadwal penghentian 171
Kebijakan Support untuk fungsi Lambda 171
Support kebijakan untuk AWS IoT Penguji Perangkat untuk AWS IoT Greengrass V1 172
Akhir jadwal pemeliharaan 172
Akhir pemeliharaan untuk perangkat lunak AWS IoT Greengrass Core v1.x gambar Docker 43
Akhir pemeliharaan untuk perangkat lunak AWS IoT Greengrass inti v1.x APT repositori 174
Akhir pemeliharaan untuk perangkat lunak AWS IoT Greengrass Core v1.11.x Snap 174
Memulai dengan AWS IoT Greengrass 175
Pilih cara memulai 175
Persyaratan 178
Buat Akun AWS 179
Mendaftar Akun AWS 180
Membuat pengguna administratif 180
Quick start: penyiapan perangkat Greengrass 181
Persyaratan 182
Jalankan penyiapan perangkat Greengrass 183
Memecahkan masalah 186
Pilihan konfigurasi penyiapan perangkat Greengrass 187
Modul 1: Penyiapan lingkungan untuk Greengrass 197
Mengatur Raspberry Pi 198
Mengatur instans Amazon EC2 206
Mengatur perangkat lain 212
Modul 2: MenginstalAWS IoT GreengrassPerangkat lunak inti 216
PenyediaanAWS IoThal untuk digunakan sebagai inti Greengrass 216
Buat grup Greengrass s s s s s s s s s s 220
Instal dan jalankanAWS IoT Greengrasspada perangkat inti 221
Modul 3 (bagian 1): Fungsi Lambda padaAWS IoT Greengrass 228
Buat dan paketkan fungsi Lambda 228
Konfigurasikan fungsi Lambda untuk AWS IoT Greengrass 233
Men-deploy konfigurasi cloud ke perangkat core 236
Verifikasi fungsi Lambda berjalan pada perangkat core 238
Modul 3 (bagian 2): Lambda berfungsi padaAWS IoT Greengrass 239

Buat dan paket fungsi Lambda 239
Mengonfigurasi fungsi Lambda berumur panjang untuk AWS IoT Greengrass 243
Uji fungsi Lambda berumur panjang 244
Uji fungsi Lambda sesuai permintaan 247
Modul 4: Berinteraksilah dengan perangkat klien dalamAWS IoT Greengrasskelompok 251
Membuat perangkat klien dalamAWS IoT Greengrasskelompok 253
Konfigurasi langganan 256
Instal AWS IoT Device SDK untuk Python 257
Uji komunikasi 263
Modul 5: Berinteraksi dengan bayangan perangkat 268
Konfigurasikan perangkat dan langganan 269
Unduh file yang diperlukan 272
Uji komunikasi (sinkronisasi perangkat dinonaktifkan) 272
Uji komunikasi (sinkronisasi perangkat diaktifkan) 276
Modul 6: Mengakses lainnyaAWSjasa 277
Mengonfigurasi peran grup 279
Buat fungsi Lambda dengan konsol 281
Konfigurasi langganan 284
Uji komunikasi 285
Modul 7: Mensimulasikan integrasi keamanan perangkat keras 287
Pasang SoftHSM 288
Konfigurasi SoftHSM 288
Mengimpor kunci privat 289
Konfigurasikan core Greengrass 291
Uji konfigurasi 294
Lihat juga 295
Perbarui OTA AWS IoT Greengrass Perangkat lunak Core 296
Persyaratan 296
Izin IAM untuk pembaruan OTA 298
Pertimbangan-pertimbangan 300
Agen pembaruan OTA Greengrass 301
Integrasi dengan sistem init 302
Respawn terkelola dengan pembaruan OTA 302
Buat pembaruan OTA 304
API CreateSoftwareUpdateJob 308
Men-deploy AWS IoT Greengrass grup 311

Men-deploy grup (konsol) 312
Men-deploy grup (API) 314
Mendapatkan ID grup 315
Ringkasan tentang model objek grup 316
Grup 317
Versi grup 317
Komponen grup 318
Memperbarui grup 319
Lihat juga 320
Dapatkan notifikasi deployment 321
Peristiwa perubahan status deployment grup 322
Prasyarat untuk membuat EventBridge aturan 323
Konfigurasikan pemberitahuan deployment (konsol) 324
Konfigurasikan pemberitahuan deployment (CLI) 325
Konfigurasikan pemberitahuan deployment (AWS CloudFormation) 326
Lihat juga 326
Atur ulang deployment 326
Atur ulang deployment dari AWS IoT konsol 327
Atur ulang deployment dengan AWS IoT Greengrass API 328
Lihat juga 329
Buat deployment massal 329
Prasyarat 330
Buat dan unggah file input deployment massal 330
Membuat dan mengonfigurasi peran eksekusi IAM untuk deployment massal 333
Izinkan akses peran eksekusi ke Bucket S3 Anda 335
Men-deploy grup 337
Uji deployment 340
Pemecahan masalah deployment massal 341
Lihat juga 344
Jalankan fungsi Lambda lokal 345
SDK 346
Memigrasi fungsi Lambda berbasis cloud 349
Fungsi referensi dengan alias atau versi 350
Mengontrol eksekusi fungsi Greengrass Lambda 351
Pengaturan konfigurasi grup khusus 351
Menjalankan fungsi Lambda sebagai root 355

Pertimbangan ketika memilih fungsi Lambda kontainerisasi 357
Mengatur identitas akses default untuk fungsi Lambda dalam grup 361
Pengaturan kontainerisasi default untuk fungsi Lambda dalam grup 362
Arus komunikasi 363
Komunikasi menggunakan pesan MQTT 363
Arus komunikasi lainnya 364
Mengambil input topik (atau subjek) 365
Konfigurasi siklus hidup 367
Lambda yang dapat dieksekusi 368
Buat Lambda yang dapat dieksekusi 370
Jalankan AWS IoT Greengrass di kontainer Docker 371
Prasyarat 373
Dapatkan citra kontainer AWS IoT Greengrass dari Amazon ECR 374
Membuat dan mengonfigurasi grup Greengrass dan core 378
Jalankan AWS IoT Greengrass secara lokal 378
Mengonfigurasi kontainerisasi "Tanpa kontainer" untuk grup 382
Men-deploy fungsi Lambda ke kontainer Docker 382
(Opsional) Men-deploy perangkat klien yang berinteraksi dengan Greengrass berjalan dalam kontainer Docker 383
Menghentikan kontainer Docker AWS IoT Greengrass ini 383
Penyelesaian masalah AWS IoT Greengrass di kontainer Docker 383
Akses sumber daya lokal 387
Jenis sumber daya yang mendukung 387
Persyaratan 389
Sumber daya volume di bawah direktori /proc 389
Izin akses file pemilik grup 390
Lihat juga 390
Menggunakan CLI 390
Buat sumber daya lokal 391
Buat fungsi Greengrass 393
Tambahkan fungsi Lambda ke grup 394
Memecahkan masalah 397
Menggunakan konsol 398
Prasyarat 398
Buat paket deployment fungsi Lambda 399
Buat dan publikasi fungsi Lambda 400

Menambahkan fungsi Lambda ke grup 403
Menambahkan sumber daya lokal ke grup 403
Menambahkan langganan ke grup 404
Men-deploy grup 405
Uji akses sumber daya lokal 407
Tampilkan inferensi machine learning 409
Bagaimana inferensi ML AWS IoT Greengrass bekerja 409
Sumber daya machine learning 410
Sumber model yang didukung 410
Persyaratan 413
Waktu aktif dan perpustakaan untuk inferensi ML 413
Waktu aktif deep learning SageMaker Neo 414
Versioning MXNet 414
MXNet pada Raspberry P 414
TensorFlow model melayani keterbatasan pada Raspberry Pi 415
Mengakses sumber daya machine learning 415
Izin akses untuk sumber daya machine learning 416
Mendefinisikan izin akses untuk fungsi Lambda (konsol) 418
Mendefinisikan izin akses untuk fungsi Lambda (API) 419
Mengakses sumber daya machine learning dari kode fungsi Lambda 422
Memecahkan masalah 423
Lihat juga 425
Cara mengonfigurasi inferensi machine learning 425
Prasyarat 426
Mengonfigurasi Raspberry Pi 427
Instal kerangka kerja MXNet 429
Buat paket model 429
Buat dan publikasikan fungsi Lambda 430
Menambahkan fungsi Lambda ke grup 433
Menambahkan sumber daya ke grup 435
Menambahkan langganan ke grup 438
Men-deploy grup 438
Tes aplikasi 440
Langkah selanjutnya 443
Mengonfigurasi Intel Atom 443
Mengonfigurasi NVIDIA Jetson TX2 447

Cara mengonfigurasi kesimpulan machine learning yang dioptimalkan 451
Prasyarat 426
Mengonfigurasi Raspberry Pi 453
PasangNeo deep learning deep learning 455
Buat fungsi inferensi Lambda 456
Tambahkan fungsi Lambda ke grup 459
Tambahkan sumber daya model NEO-dioptimalkan ke grup 461
Tambahkan sumber daya perangkat kamera ke grup 464
Tambahkan langganan ke grup 465
Men-deploy grup 465
Uji contoh 466
Mengkonfigurasi Atom Intel 467
Mengkonfigurasi NVIDIA Jetson TX2 470
Pemecahan masalah AWS IoT Greengrass inferensi ML 441
Langkah selanjutnya 477
Mengelola aliran data 478
alur kerja manajemen pengaliran 479
Persyaratan 481
Keamanan data 482
Keamanan data lokal 482
Autentikasi klien 483
Lihat juga 483
Konfigurasi pengelola pengaliran 484
Parameter pengelola pengaliran 484
Konfigurasikan pengaturan (konsol) 487
Konfigurasi pengaturan (CLI) 490
Lihat juga 500
Gunakan StreamManagerClient untuk bekerja dengan aliran 500
Membuat pengaliran pesan 502
Tambahkan pesan 506
Baca pesan 512
Daftar aliran 515
Jelaskan aliran pesan 517
Perbarui aliran pesan 519
Hapus aliran pesan 523
Lihat juga 524

Konfigurasi ekspor untuk tujuan AWS Cloud yang didukung 525
Ekspor aliran data (konsol) 541
Prasyarat 542
Buat paket deployment fungsi Lambda 545
Buat fungsi Lambda 548
Tambahkan fungsi ke grup 550
Mengaktifkan pengelola pengaliran 551
Konfigurasikan logging lokal 551
Men-deploy grup 552
Uji aplikasi 553
Lihat juga 555
Ekspor aliran data (CLI) 555
Prasyarat 556
Buat paket deployment fungsi Lambda 558
Buat fungsi Lambda 562
Buat definisi fungsi dan versi 564
Buat definisi pencatat dan versi 566
Dapatkan ARN versi definisi core Anda 567
Buat versi grup 568
Buat deployment 569
Uji aplikasi 570
Lihat juga 572
Men-deploy rahasia ke core 573
Enkripsi rahasia 574
Persyaratan 576
Tentukan kunci privat untuk enkripsi rahasia 577
Mengizinkan AWS IoT Greengrass untuk mendapatkan nilai rahasia 578
Lihat juga 580
Bekerja dengan sumber daya rahasia 580
Membuat dan mengelola rahasia 580
Menggunakan rahasia lokal 585
Cara membuat sumber daya rahasia (konsol) 588
Prasyarat 590
Buat rahasia Secrets Manager 590
Menambahkan sumber daya rahasia ke grup 591
Buat paket deployment fungsi Lambda 592

Buat fungsi Lambda 594
Menambahkan fungsi ke grup 596
Lampirkan sumber daya rahasia ke fungsi Lambda 598
Menambahkan langganan ke grup 598
Men-deploy grup 599
Tes fungsi Lambda 600
Lihat juga 601
Integrasikan dengan layanan dan protokol menggunakan konektor 602
Persyaratan 603
Menggunakan konektor Greengrass 604
Parameter konfigurasi 606
Parameter yang digunakan untuk mengakses sumber daya grup 606
Memperbarui parameter konektor 607
Input dan output 607
Topik input 608
Dukungan kontainerisasi 609
Versi upgrade konektor 609
Mencatat 610
AWS-disediakan konektor Greengrass 611
CloudWatch Metrik 614
Pertahanan Perangkat 630
Deployment aplikasi Docker 637
IoT Analytics 680
Adaptor Protokol IP Ethernet IoT 696
IoT SiteWise 701
Kinesis Firehose 717
Umpan balik ML 735
Klasifikasi Citra ML 753
Deteksi Objek ML 779
Adaptor Protokol Modbus-RTU 796
Adaptor Protokol Modbus-TCP 815
Raspberry pi GPIO 820
Aliran Serial 831
ServiceNow MetricBase Integrasi 844
SNS 860
Integrasi Splunk 871

Notifikasi Twilio ... 885
Memulai dengan konektor (konsol) ... 902
Prasyarat ... 904
Buat rahasia Secrets Manager ... 904
Tambahkan sumber daya rahasia ke grup ... 905
Tambahkan konektor ke grup ... 906
Buat paket deployment fungsi Lambda ... 907
Buat fungsi Lambda ... 909
Tambahkan fungsi ke grup ... 911
Menambahkan langganan ke grup ... 911
Men-deploy grup ... 912
Pengujian solusi ... 913
Lihat juga ... 914
Memulai dengan konektor (CLI) ... 915
Prasyarat ... 917
Buat rahasia Secrets Manager ... 918
Buat definisi sumber daya dan versi ... 918
Buat definisi konektor dan versi ... 919
Buat paket deployment fungsi Lambda ... 921
Buat fungsi Lambda ... 922
Buat definisi fungsi dan versi ... 924
Buat definisi langganan dan versi ... 925
Buat versi grup ... 927
Buat deployment ... 929
Uji solusi ... 930
Lihat juga ... 931
Greengrass Discovery RESTful API ... 932
Permintaan ... 932
Response ... 933
Otorisasi Discovery ... 934
Contoh menemukan dokumen respon ... 934
Keamanan ... 937
Ikhtisar AWS IoT Greengrass keamanan ... 938
Alur kerja koneksi perangkat ... 939
Mengkonfigurasi keamanan AWS IoT Greengrass ... 940
Keamanan utama ... 941

Langganan yang dikelola di dalam alur kerja pesan MQTT 944
TLS cipher suite mendukung 944
Perlindungan data 947
Enkripsi data 948
Integrasi keamanan perangkat keras 951
Autentikasi dan otorisasi perangkat 972
Sertifikat X.509 973
AWS IoT kebijakan 975
Minimal AWS IoT kebijakan untuk perangkat core 978
Identity and access management 982
Audiens 982
Autentikasi menggunakan identitas 983
Mengelola akses menggunakan kebijakan 986
Lihat juga 989
Bagaimana AWS IoT Greengrass bekerja dengan IAM 989
Peran layanan Greengrass 998
Peran grup Greengrass 1006
Cross-service bingung wakil pencegahan 1016
Contoh kebijakan berbasis identitas 1017
Pemecahan masalah identitas dan akses 1021
Validasi Kepatuhan 1024
Ketahanan 1025
Keamanan infrastruktur 1026
Analisis konfigurasi dan kelemahan 1027
Titik akhir VPC (AWS PrivateLink) 1028
Pertimbangan untuk VPC endpoint AWS IoT Greengrass 1029
Buat VPC endpoint antarmuka untukAWS IoT GreengrassOperasi bidang kendali 1029
Membuat kebijakan VPC endpoint untuk AWS IoT Greengrass 1029
Praktik terbaik keamanan 1030
Berikan izin minimum yang memungkinkan 1030
Jangan me-hardcode kredensial dalam fungsi Lambda 1031
Jangan log informasi sensitif 1031
Buat langganan yang ditargetkan 1031
Sinkronkan jam perangkat Anda 1032
Mengelola autentikasi perangkat dengan core Greengrass 1032
Lihat juga 1033

Pencatatan dan pemantauan 1034
Alat pemantauan 1034
Lihat juga 1035
Pemantauan dengan AWS IoT Greengrass log 1035
Mengakses Log CloudWatch 1035
Mengakses log sistem file 1037
Konfigurasi pencatatan default 1038
Konfigurasi pencatatan untuk AWS IoT Greengrass 1039
Batasan pencatatan 1042
CloudTrail log 1044
Mencatat log panggilan API AWS IoT Greengrass dengan AWS CloudTrail 1044
AWS IoT Greengrassinformasi di CloudTrail 1044
Memahami entri file log AWS IoT Greengrass 1045
Lihat juga 1049
Mengumpulkan data telemetri kondisi sistem 1049
Mengonfigurasi pengaturan telemetri 1052
Berlangganan untuk menerima data telemetri 1056
Penyelesaian Masalah AWS IoT Greengrass telemetri 1063
Memanggil API pemeriksaan kondisi lokal 1063
Dapatkan informasi kondisi untuk semua pekerja 1063
Dapatkan informasi kondisi tentang pekerja tertentu 1065
Informasi kondisi pekerja 1067
Menandai sumber daya Greengrass Anda 1071
Dasar-dasar tanda 1071
Dukungan penandaan (konsol) 1071
Dukungan penandaan (API) 1072
Menggunakan tanda dengan kebijakan IAM 1074
Contoh kebijakan IAM 1075
Lihat juga 1077
AWS CloudFormation dukungan untuk AWS IoT Greengrass 1078
Buat sumber daya 1078
Terapkan sumber daya 1079
Contoh Templat 1080
Didukung Wilayah AWS s 1093
Menggunakan AWS IoT Penguji Perangkat untuk AWS IoT Greengrass V1 1094
AWS IoT Greengrass suite kualifikasi 1094

Rangkaian pengujian khusus 1095
Versi yang didukung AWS IoT Penguji Perangkat untuk AWS IoT Greengrass V1 1095
Versi IDT yang tidak didukung untuk AWS IoT Greengrass 1096
Gunakan IDT untuk menjalankan AWS IoT Greengrass suite kualifikasi 1101
Versi rangkaian pengujian 1102
Deskripsi grup tes 1103
Prasyarat 1108
Konfigurasikan perangkat Anda untuk menjalankan tes IDT 1117
Konfigurasikan pengaturan IDT 1140
Jalankan AWS IoT Greengrass suite kualifikasi 1155
Memahami hasil dan mencatat 1160
Gunakan IDT untuk mengembangkan dan menjalankan rangkaian pengujian Anda sendiri 1164
Unduh versi terbaru IDT untuk AWS IoT Greengrass 1108
Alur kerja pembuatan rangkaian uji 1165
Tutorial: Bangun sampel rangkaian tes IDT 1165
Tutorial: Kembangkan rangkaian tes IDT sederhana 1171
Buat file konfigurasi rangkaian tes IDT 1180
Konfigurasikan mesin status IDT 1188
Buat executable uji kasus IDT 1212
Gunakan konteks IDT 1219
Mengonfigurasi pengaturan untuk test runner 1224
Debug dan jalankan rangkaian tes kustom 1235
Tinjau hasil tes IDT dan log 1238
Metrik penggunaan IDT 1245
IDT untuk AWS IoT Greengrass pemecahan masalah 1251
Kode eror 1251
Menyelesaikan IDT untuk AWS IoT Greengrass eror 1274
Support kebijakan untuk AWS IoT Penguji Perangkat untuk AWS IoT Greengrass V1 1280
Memecahkan masalah 1281
AWS IoT Greengrass Masalah core 1281
Kesalahan: File konfigurasi hilang CaPath, CertPath atau KeyPath. <pid>Proses daemon Greengrass dengan [pid =] mati. 1283
Error: Gagal mengurai/<greengrass-root> /config/config.json. 1284
Kesalahan: Terjadi kesalahan saat membuat konfigurasi TLS: UrisCheme ErrUnknown 1284
Error: Waktu aktif gagal untuk mulai: tidak dapat mulai pekerja: pengujian kontainer habis waktu. 1285

<address>Kesalahan: Gagal memanggil PutLogEvents di Cloudwatch lokal, LogGroup:/GreengrassSystem/connection_manager, kesalahan:: permintaan kirim gagal disebabkan oleh: Posting http RequestError://<path>/cloudwatch/logs/: dial tcp: getsockopt: koneksi ditolak, respons: {}. 1285
<region><account-id><function-name><version><file-name>Kesalahan: Tidak dapat membuat server karena: gagal memuat grup: chmod/<greengrass-root>/ggc/deployment/lambda/arn:aws:lambda: ::function::/: tidak ada file atau direktori seperti itu. 1286
Perangkat lunak Core AWS IoT Greengrass tidak mulai setelah Anda berubah dari berjalan tanpa kontainerisasi untuk berjalan dalam kontainer Greengrass. 1286
error: Ukuran spool harus setidaknya 262144 byte. 1286
Kesalahan: [KESALAHAN]-kesalahan olahpesan cloud: Terjadi kesalahan ketika mencoba menerbitkan pesan. {"errorString": "timed out operasi"} 1287
<version>Kesalahan: container_linux.go: 344: memulai proses kontainer menyebabkan “process_linux.go: 424: container init disebabkan\" rootfs_linux.go:64: mounting\\\” /greengrass/ggc/socket/greengrass_ipc.sock\\\” ke rootfs\\\” /greengrass/ggc/packageses/ /rootfs/merged\\\” di\\\” /greengrass_ipc.sock\\\” disebabkan\\\ "stat /greengrass/ggc/socket/greengrass_ipc.sock: izin ditolak\\\"\ "”. 1288
error: Greengrass daemon berjalan dengan PID: <process-id>. Beberapa komponen sistem gagal untuk mulai. Periksa 'runtime.catatan' untuk error. 1288
Bayangan perangkat tidak sinkron dengan cloud. 1022
ERROR: tidak dapat menerima koneksi TCP. menerima tcp [::]:8000 accept4: terlalu banyak file terbuka. 1289
Error: error eksekusi waktu aktif: tidak dapat memulai kontainer lambda. container_linux.go: 259: memulai proses kontainer disebabkan "process_linux.go:345: container init disebabkan \"rootfs_linux.go:50: mempersiapkan rootfs disebabkan \\\ "izin ditolak\\\"\"". 1289
Peringatan: [PERINGATAN] - [5] GK Remote: Kesalahan mengambil data kunci publik: ErrPrincipalNotConfigured: kunci pribadi untuk tidak MqttCertificate disetel. 1289
<account-id><role-name><region>Kesalahan: Izin ditolak saat mencoba menggunakan peran arn:aws:iam: ::role/ untuk mengakses url s3 https://-greengrass-updates.s3. <region><architecture><distribution-version>.amazonaws.com/core/ /greengrass-core- .tar.gz. 1022
Core AWS IoT Greengrass dikonfigurasikan untuk menggunakan proksi jaringan dan fungsi Lambda Anda tidak dapat membuat koneksi keluar. 1290
Core adalah dalam loop connect-disconnect yang tak terbatas. File runtime.log berisi serangkaian kontinyu entri menghubungkan dan memutuskan. 1291

Error: tidak dapat memulai kontainer lambda. container_linux.go:259: memulai proses kontainer disebabkan "process_linux.go:345: container init disebabkan \"rootfs_linux.go: 62: mounting\\\ "proc\\\" to rootfs\\\" 1292
[ERROR]-error ekseskusi waktu aktif: tidak dapat memulai kontainer lambda. {"errorString": "gagal menginisialisasi pemasangan kontainer: gagal untuk menutupi root greengrass di overlay dir atas: gagal membuat perangkat mask pada direktori <ggc-path>: file ada"} 1292
[ERROR] -Penerapan gagal. {"deploymentID": <deployment-id>"", “errorString”: “proses pengujian kontainer dengan <pid>pid gagal: status proses kontainer: status keluar 1"} 1293
Error: [ERROR]-error eksekusi waktu aktif: tidak dapat memulai kontainer lambda. {"ErrorString": "gagal menginisialisasi pemasangan kontainer: gagal membuat overlay fs untuk kontainer: pemasangan overlay pada /greengrass/ggc/packages/ <ggc-version> / rootfs/digabung gagal: gagal untuk me-mount dengan sumber args=\"no_source \" dest=\"/greengrass/ggc/packages/<ggc-version>/rootfs/merged\" fstype=\"overlay\" flags= \"0\" data=\"lowerdir=/greengrass/ggc/packages/<ggc-version>/dns:/,upperdir=/greengrass/ggc/packages/<ggc-version>/rootfs/upper,workdir=/greengrass/ggc/packages/<ggc-version>/rootfs/work\": terlalu banyak tingkatan tautan simbolik"} 1294
Error: [DEBUG]-Gagal untuk mendapatkan rute. Membuang pesan. 1295
Error: [Errno 24] Terlalu banyak membuka <lambda-function>, [Errno 24] Terlalu banyak membuka file 1295
Kesalahan: server ds gagal mulai mendengarkan soket: dengarkan unix <ggc-path>/ggc/socket/greengrass_ipc.sock: bind: argumen tidak valid 1295
[INFO] (Mesin Fotokopi) aws.greengrass. StreamManager: stdout. Disebabkan oleh: com.fasterxml.jackson.databind. JsonMappingException: Instan melebihi minimum atau maksimum instan 1296
GPG error: https://dnw9lb6lzp2d8.cloudfront.net stable InRelease: The following signatures were invalid: EXPKEYSIG 68D644ABD2327D47 AWS Greengrass Master Key 1296
Masalah deployment 1297
Deployment Anda saat ini tidak bekerja dan Anda ingin kembali ke deployment kerja sebelumnya. 1298
Anda melihat 403 error terlarang pada deployment di catatan. 1300
Terjadi ConcurrentDeployment kesalahan saat Anda menjalankan perintah create-deployment untuk pertama kalinya. 1301
Error: Greengrass tidak berwenang untuk menganggap peran layanan yang terkait dengan akun ini, atau error: Gagal: peran layanan TES tidak terkait dengan akun ini. 1022

Error: tidak dapat menjalankan langkah pengunduhan dalam deployment. error saat mengunduh: error saat mengunduh file definisi Grup:... x509: sertifikat telah kedaluwarsa atau belum valid 1301
Deployment tidak selesai. 1301
Kesalahan: Tidak dapat menemukan executable java atau java8, atau kesalahan: Penerapan <deployment-id>tipe NewDeployment untuk grup <group-id>gagal kesalahan: pekerja dengan <worker-id>gagal menginisialisasi dengan alasan Versi Java yang diinstal harus lebih besar dari atau sama dengan 8 1302
Deployment tidak selesai, dan runtime.catatan berisi beberapa entri "tunggu 1 detik untuk kontainer untuk berhenti". 1303
Deployment tidak selesai, dan `runtime.log` berisi "[KESALAHAN]-Greengrass deployment error: gagal melaporkan status deployment kembali ke cloud {"deploymentId": "<deployment-id>", "errorString": "Failed to initiate PUT, endpoint: https://<deployment-status>, error: Put https://<deployment-status>: proxyconnect tcp: x509: certificate signed by unknown authority"}" 1303
<path>Kesalahan: <deployment-id>Penerapan tipe NewDeployment untuk grup <group-id>gagal kesalahan: Kesalahan saat memproses. konfigurasi grup tidak valid: 112 atau [1190] tidak memiliki izin rw pada file:. 1304
Kesalahan: < list-of-function-arns > dikonfigurasi untuk berjalan sebagai root tetapi Greengrass tidak dikonfigurasi untuk menjalankan fungsi Lambda dengan izin root. 1304
Kesalahan: <deployment-id>Penerapan tipe NewDeployment untuk grup <group-id>gagal kesalahan: Kesalahan penerapan Greengrass: tidak dapat menjalankan langkah unduhan dalam penerapan. kesalahan saat memproses: tidak dapat memuat file grup yang diunduh: tidak dapat menemukan UID berdasarkan nama pengguna, UserName: ggc_user: user: unknown user ggc_user. 1305
Error: [ERROR]-error eksekusi waktu aktif: tidak dapat meluncurkan kontainer lambda. {"ErrorString": "gagal menginisialisasi pemasangan kontainer: gagal untuk menutupi root greengrass di overlay dir atas: gagal membuat perangkat mask pada direktori <ggc-path>: file ada"} 1305
Kesalahan: Penerapan <deployment-id>tipe NewDeployment untuk grup <group-id>gagal kesalahan: proses mulai gagal: container_linux.go: 259: memulai proses kontainer menyebabkan “process_linux.go: 250: menjalankan exec setns process for init caused\" wait: no child processes\ ""”. 1306
<host-prefix>Kesalahan: [PERINGATAN] -MQTT [klien] panggil tcp: lookup -ats.iot.<region>.amazonaws.com: tidak ada host seperti itu... [ERROR]-Greengrass deployment

error: gagal melaporkan status deployment kembali ke cloud ... net/http: permintaan dibatalkan saat menunggu koneksi (Client.Timeout terlampaui saat menunggu header) 1306
Buat grup dan buat masalah fungsi 1307
Kesalahan: Konfigurasi IsolationMode '' Anda untuk grup tidak valid. 1307
Kesalahan: Konfigurasi 'IsolationMode' Anda untuk fungsi dengan arn <function-arn>tidak valid. 1308
Kesalahan: MemorySize konfigurasi untuk fungsi dengan arn <function-arn>tidak diperbolehkan di IsolationMode =NoContainer. 1308
Kesalahan: Akses konfigurasi Sysfs untuk fungsi dengan arn <function-arn>tidak diperbolehkan di =. IsolationMode NoContainer 1308
Kesalahan: MemorySize konfigurasi untuk fungsi dengan arn <function-arn>diperlukan di IsolationMode =GreengrassContainer. 1308
Kesalahan: Fungsi <function-arn>mengacu pada sumber daya tipe <resource-type>yang tidak diperbolehkan di IsolationMode =NoContainer. 1309
Error: Konfigurasi eksekusi untuk fungsi dengan arn <function-arn> tidak diizinkan. 1309
Masalah Penemuan 1309
Error: Perangkat adalah anggota dari terlalu banyak grup, perangkat mungkin tidak berada di lebih dari 10 grup 1309
Masalah sumber daya machine learning 1310
InvalidML ModelOwner - GroupOwnerSetting disediakan dalam sumber daya model ML, tetapi GroupOwner atau GroupPermission tidak ada 424
NoContainer fungsi tidak dapat mengonfigurasi izin saat melampirkan sumber daya Machine Learning. <function-arn>mengacu pada sumber daya Machine Learnin <resource-id>dengan izin <ro/rw> dalam kebijakan akses sumber daya. 424
Fungsi <function-arn>mengacu pada sumber daya Machine Learning <resource-id>dengan izin yang hilang di keduanya ResourceAccessPolicy dan sumber daya OwnerSetting. 424
Fungsi <function-arn>mengacu pada sumber daya Machine Learning <resource-id>dengan izin\ "rw\”, sedangkan pengaturan pemilik sumber daya GroupPermission hanya mengizinkan\ "ro\”. 424
NoContainer Fungsi <function-arn>mengacu pada sumber daya jalur tujuan bersarang. 425
Lambda <function-arn> mendapatkan akses ke sumber daya <resource-id> dengan berbagi id pemilik grup yang sama 425
AWS IoT Greengrass core dalam masalah Docker 1312
Kesalahan: Opsi tidak diketahui: -no-include-email. 383
Peringatan: IPv4 dinonaktifkan. Jaringan tidak akan bekerja. 384
Error: Firewall memblokir file berbagi antara windows dan kontainer. 384

Kesalahan: Terjadi kesalahan (AccessDeniedException) saat memanggil GetAuthorizationToken operasi: User: arn:aws:iam: ::user/ <account-id><user-name>tidak diizinkan untuk melakukan: ecr: on resource: * GetAuthorizationToken 384
Error: Tidak dapat membuat kontainer untuk greengrass layanan: konflik. Nama kontainer “/aws-iot-greengrass" sudah digunakan. 1314
Error: [FATAL]-Gagal untuk mereset thread mount namespace karena error tak terduga: "operasi tidak diizinkan". Untuk menjaga konsistensi, GGC akan macet dan harus dimulai ulang secara manual. 1314
Pemecahan masalah dengan catatan 1315
Pemecahan masalah penyimpanan 1316
Pemecahan masalah pesan 1316
Memecahkan masalah timeout sinkronisasi bayangan 1317
Periksa Re: AWS Posting 1318
Riwayat dokumen 1319
Pembaruan sebelumnya 1342

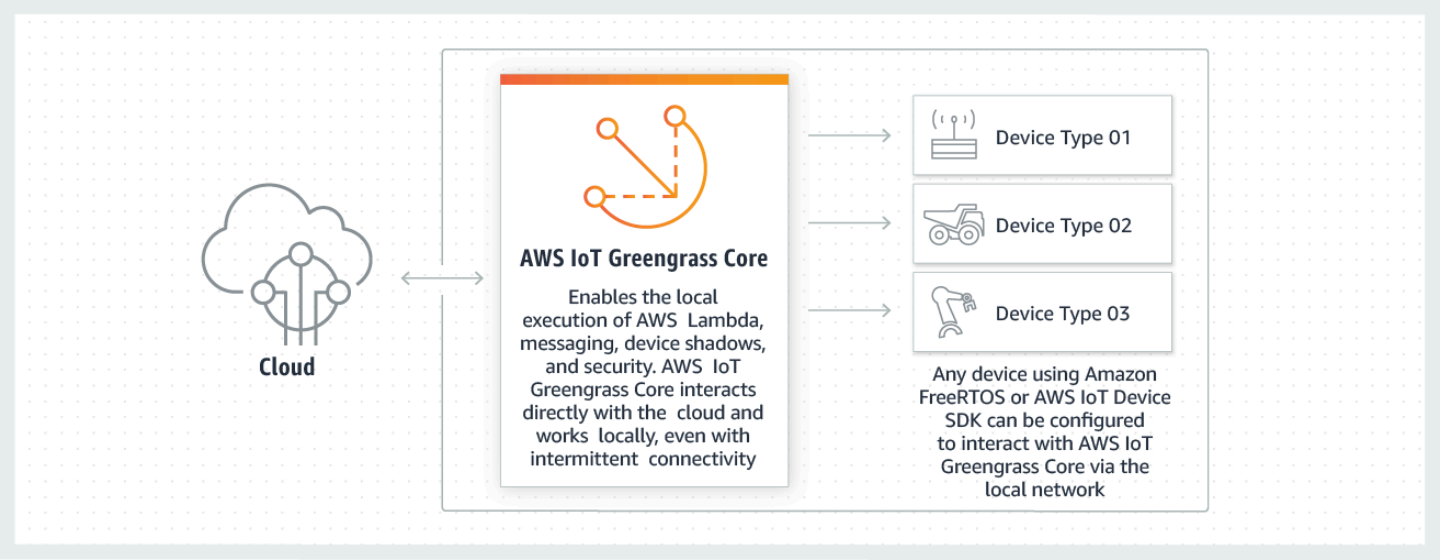
AWS IoT Greengrass Version 1 memasuki fase umur panjang pada 30 Juni 2023. Untuk informasi selengkapnya, lihat kebijakan AWS IoT Greengrass V1 pemeliharaan. Setelah tanggal ini, tidak AWS IoT Greengrass V1 akan merilis pembaruan yang menyediakan fitur, penyempurnaan, perbaikan bug, atau patch keamanan. Perangkat yang berjalan AWS IoT Greengrass V1 tidak akan terganggu dan akan terus beroperasi dan terhubung ke cloud. Kami sangat menyarankan Anda bermigrasi ke AWS IoT Greengrass Version 2, yang menambahkan fitur baru yang signifikan dan dukungan untuk platform tambahan.

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.

Apakah AWS IoT Greengrass itu?

AWS IoT Greengrass adalah perangkat lunak yang memperluas kemampuan cloud ke perangkat lokal. Hal ini memungkinkan perangkat untuk mengumpulkan dan menganalisis data lebih dekat ke sumber informasi, bereaksi secara mandiri terhadap acara lokal, dan berkomunikasi secara aman satu sama lain di jaringan lokal. Perangkat lokal juga dapat berkomunikasi secara aman dengan AWS IoT Core dan mengekspor data IoT ke AWS Cloud. AWS IoT Greengrass developer dapat menggunakan AWS Lambda fungsi dan prebuilt konektor untuk membuat aplikasi nirserver yang di-deploy ke perangkat untuk eksekusi lokal.

Diagram berikut menunjukkan arsitektur dasar dari AWS IoT Greengrass.



AWS IoT Greengrass memungkinkan pelanggan untuk membangun perangkat IoT dan logika aplikasi. Secara khusus, AWS IoT Greengrass menyediakan pengelolaan logika aplikasi berbasis cloud yang berjalan di perangkat. Fungsi Lambda yang di-deploy secara lokal dan konektor dipicu oleh acara lokal, pesan dari cloud, atau sumber lainnya.

Pada AWS IoT Greengrass, perangkat berkomunikasi dengan aman di jaringan lokal dan bertukar pesan satu sama lain tanpa harus connect ke cloud. AWS IoT Greengrass menyediakan manajer pesan pub/sub lokal yang secara cerdas dapat mendukung pesan jika konektivitas hilang sehingga pesan inbound dan outbound ke cloud dapat dipertahankan.

AWS IoT Greengrass melindungi data pengguna:

- Melalui autentikasi dan otorisasi perangkat yang aman.

- Melalui konektivitas aman di jaringan lokal.
- Antara perangkat lokal dan cloud.

Kredensial keamanan perangkat berfungsi dalam grup sampai mereka dicabut, bahkan jika konektivitas ke cloud terganggu, sehingga perangkat dapat terus aman berkomunikasi secara lokal.

AWS IoT Greengrassmenyediakan over-the-air pembaruan fungsi Lambda yang aman.

AWS IoT Greengrass terdiri dari:

- Distribusi perangkat lunak
  - AWS IoT Greengrass Perangkat lunak Core
  - AWS IoT Greengrass Core SDK
- Layanan cloud
  - API AWS IoT Greengrass
- Fitur
  - waktu aktif Lambda
  - Penerapan bayangan
  - Manajer pesan
  - Manajemen grup
  - Layanan Penemuan
  - O agen ver-the-air pembaruan
  - Pengelola aliran
  - Akses sumber daya lokal
  - Inferensi machine learning lokal
  - Secrets Manager lokal
  - Konektor dengan integrasi bawaan dalam layanan, protokol, dan perangkat lunak

Topik

- AWS IoT Greengrass Perangkat lunak Core
- AWS IoT Greengrass Grup
- Perangkat di AWS IoT Greengrass
- SDK

- Platform dan persyaratan yang didukung
- AWS IoT Greengrass unduh
- Kami ingin mendengar pendapat Anda
- Instal AWS IoT Greengrass perangkat lunak Core
- Mengonfigurasi AWS IoT Greengrass core

AWS IoT Greengrass Perangkat lunak Core

Perangkat lunak Core AWS IoT Greengrass menyediakan fungsi berikut:

- Deployment dan menjalankan lokal konektor dan fungsi Lambda.
- Memproses aliran data secara lokal dengan ekspor otomatis ke AWS Cloud.
- Olahpesan MQTT melalui jaringan lokal antara perangkat, konektor, dan fungsi Lambda menggunakan langganan terkelola.
- Olahpesan MQTT antara AWS IoT dan perangkat, konektor, dan fungsi Lambda dapat menggunakan langganan terkelola.
- Koneksi aman antara perangkat dan AWS Cloud menggunakan autentikasi dan otorisasi perangkat.
- Sinkronisasi bayangan lokal perangkat. Bayangan dapat dikonfigurasi untuk disinkronkan dengan AWS Cloud.
- Akses terkontrol ke perangkat lokal dan sumber daya volume.
- Deployment model machine learning dengan pelatihan cloud untuk menjalankan inferensi lokal.
- Deteksi alamat IP otomatis yang mengaktifkan perangkat untuk menemukan perangkat Core Greengrass.
- Deployment pusat dar konfigurasi grup yang baru atau diperbarui. Setelah data konfigurasi diunduh, perangkat core di-restart secara otomatis.
- Pembaruan perangkat lunak over-the-air (OTA) yang aman dari fungsi Lambda yang ditentukan pengguna.
- Penyimpanan rahasia lokal yang aman dan terenkripsi dan akses yang dikendalikan oleh konektor dan fungsi Lambda.

AWS IoT Greengrass instans core dikonfigurasi melalui AWS IoT Greengrass API yang membuat dan memperbarui definisi grup AWS IoT Greengrass yang disimpan di cloud.

AWS IoT Greengrass Versi perangkat lunak core

AWS IoT Greengrass menyediakan beberapa opsi untuk menginstal AWS IoT Greengrass perangkat lunak Core, termasuk file unduhan tar.gz, skrip quick start, dan `apt` intalasi pada platform Debian yang mendukung. Untuk informasi selengkapnya, lihat the section called “Instal AWS IoT Greengrass perangkat lunak Core”.

Tab berikut menjelaskan apa yang baru dan diubah di AWS IoT Greengrass versi perangkat lunak core.

GGC v1.11

1.11.6

Perbaikan bug dan peningkatan:

- Peningkatan ketahanan jika kehilangan daya mendadak terjadi selama penyebaran.
- Memperbaiki masalah di mana kerusakan data manajer aliran dapat mencegah perangkat lunak AWS IoT Greengrass Core dimulai.
- Memperbaiki masalah di mana perangkat klien baru tidak dapat terhubung ke inti dalam skenario tertentu.
- Memperbaiki masalah di mana nama aliran pengelola aliran tidak dapat berisi `.log`.

1.11.5

Perbaikan bug dan peningkatan:

- Peningkatan performa umum dan perbaikan bug.

1.11.4

Perbaikan bug dan peningkatan:

- Memperbaiki masalah dengan pengelola aliran yang mencegah peningkatan ke perangkat lunak AWS IoT Greengrass Core v1.11.3. Jika Anda menggunakan stream manager untuk mengekspor data ke cloud, Anda sekarang dapat menggunakan pembaruan OTA untuk memutakhirkan versi v1.x sebelumnya dari perangkat lunak AWS IoT Greengrass Core ke v1.11.4.
- Peningkatan performa umum dan perbaikan bug.

1.11.3

Perbaikan bug dan peningkatan:

- Memperbaiki masalah yang menyebabkan perangkat lunak Core AWS IoT Greengrass berjalan dalam snap pada perangkat Ubuntu yang kemudian berhenti merespons setelah tiba-tiba kehilangan daya pada perangkat.
- Memperbaiki masalah yang menyebabkan keterlambatan pengiriman pesan MQTT ke fungsi Lambda berumur panjang.
- Memperbaiki masalah yang menyebabkan pesan MQTT tidak dikirim dengan benar saat nilai `maxWorkItemCount` telah diatur ke nilai yang lebih besar dari `1024`.
- Memperbaiki masalah yang menyebabkan agen pembaruan OTA mengabaikan MQTT periode `KeepAlive` yang ditentukan dalam properti `keepAlive` di `config.json`.
- Peningkatan performa umum dan perbaikan bug.

> **Important**
>
> Jika Anda menggunakan pengelola aliran untuk mengekspor data ke cloud, jangan tingkatkan ke perangkat lunak AWS IoT Greengrass Core v1.11.3 dari versi v1.x sebelumnya. Jika Anda mengaktifkan pengelola aliran untuk pertama kalinya, kami sangat menyarankan Anda menginstal versi terbaru perangkat lunak AWS IoT Greengrass Core terlebih dahulu.

1.11.1

Perbaikan bug dan peningkatan:

- Memperbaiki masalah yang menyebabkan peningkatan penggunaan memori untuk pengelola aliran.
- Memperbaiki masalah yang menyebabkan pengelola aliran mengatur ulang nomor urut aliran `0` jika perangkat inti Greengrass dimatikan lebih lama dari periode data aliran yang time-to-live ditentukan (TTL).
- Memperbaiki masalah yang mencegah pengelola aliran menghentikan upaya coba lagi dengan benar untuk mengekspor data ke AWS Cloud.

1.11.0

Fitur-fitur baru:

- Agen telemetri pada core Greengrass mengumpulkan data telemetri lokal dan menerbitkannya ke AWS Cloud. Untuk mengambil data telemetri untuk diproses lebih lanjut,

pelanggan dapat membuat EventBridge aturan Amazon dan berlangganan target. Untuk informasi lebih lanjut, lihat Mengumpulkan data telemetri kondisi sistem dari perangkat AWS IoT Greengrass core.

- API HTTP lokal mengembalikan snapshot dari keadaan saat ini dari proses pekerja lokal dimulai dengan AWS IoT Greengrass. Untuk informasi lebih lanjut, lihat Memanggil API pemeriksaan kondisi lokal.
- Sebuah Pengelola aliran secara otomatis mengekspor data ke Amazon S3 dan AWS IoT SiteWise.

  Parameter pengelola aliran baru memungkinkan Anda memperbarui aliran yang ada dan menjeda atau melanjutkan ekspor data.
- Support untuk menjalankan fungsi Python 3.8.x Lambda pada core.
- Properti `ggDaemonPort` baru di `config.json` yang digunakan untuk mengonfigurasi nomor port IPC core Greengrass. Nomor port default adalah 8000.

  Properti `systemComponentAuthTimeout` baru di `config.json` yang Anda gunakan untuk mengonfigurasi timeout untuk autentikasi IPC core Greengrass. Timeout default adalah 5000 milidetik.
- Meningkatkan jumlah maksimum perangkat AWS IoT setiap grup AWS IoT Greengrass dari 200 ke 2500.

  Meningkatkan jumlah langganan maksimum per grup dari 1000 menjadi 10000.

  Untuk informasi lebih lanjut, lihat AWS IoT Greengrass kuota dan titik akhir.

Perbaikan bug dan peningkatan:

- Optimasi umum yang dapat mengurangi pemanfaatan memori dari proses layanan Greengrass.
- Parameter konfigurasi waktu aktif baru (`mountAllBlockDevices`) memungkinkan Greengrass menggunakan ikatan pasang untuk memasang semua perangkat blok ke dalam kontainer setelah mengatur OverlayFS. Fitur ini memecahkan masalah yang menyebabkan kegagalan deployment Greengrass jika `/usr` tidak berada di bawah hierarki `/` ini.
- Memperbaiki masalah yang menyebabkan kegagalan core AWS IoT Greengrass jika `/tmp` adalah symlink.
- Memperbaiki masalah untuk membiarkan agen deployment Greengrass menghapus artefak model machine learning yang tidak digunakan dari folder `mlmodel_public` ini.

- Peningkatan performa umum dan perbaikan bug.

Extended life versions

1.10.5

Perbaikan bug dan peningkatan:

- Peningkatan performa umum dan perbaikan bug.

1.10.4

Perbaikan bug dan peningkatan:

- Memperbaiki masalah yang menyebabkan perangkat lunak Core AWS IoT Greengrass berjalan dalam snap pada perangkat Ubuntu yang kemudian berhenti merespons setelah tiba-tiba kehilangan daya pada perangkat.
- Memperbaiki masalah yang menyebabkan keterlambatan pengiriman pesan MQTT ke fungsi Lambda berumur panjang.
- Memperbaiki masalah yang menyebabkan pesan MQTT tidak dikirim dengan benar saat nilai `maxWorkItemCount` telah diatur ke nilai yang lebih besar dari `1024`.
- Memperbaiki masalah yang menyebabkan agen pembaruan OTA mengabaikan MQTT periode `KeepAlive` yang ditentukan dalam properti `keepAlive` di `config.json`.
- Peningkatan performa umum dan perbaikan bug.

1.10.3

Perbaikan bug dan peningkatan:

- Properti `systemComponentAuthTimeout` baru di `config.json` yang Anda gunakan untuk mengonfigurasi timeout untuk autentikasi IPC core Greengrass. Timeout default adalah 5000 milidetik.
- Memperbaiki masalah yang menyebabkan peningkatan penggunaan memori untuk pengelola aliran.

1.10.2

Perbaikan bug dan peningkatan:

- Properti `mqttOperationTimeout` baru di config.json yang Anda gunakan untuk mengatur timeout agar mempublikasi, berlangganan, dan berhenti berlangganan operasi di koneksi MQTT dengan AWS IoT Core.

- Peningkatan performa umum dan perbaikan bug.

1.10.1

Perbaikan bug dan peningkatan:

- Pengelola aliran lebih tahan terhadap korupsi data file.
- Memperbaiki masalah yang menyebabkan kegagalan memasang sysfs pada perangkat yang menggunakan kernel Linux 5.1 dan yang lebih baru.
- Peningkatan performa umum dan perbaikan bug.

1.10.0

Fitur-fitur baru:

- Pengelola aliran yang memproses aliran data secara lokal dan mengekspornya ke AWS Cloud secara otomatis. Fitur ini memerlukan Java 8 pada perangkat Core Greengrass. Untuk informasi selengkapnya, lihat Mengelola aliran data.
- Sebuah konektor deployment aplikasi Greengrass Docker baru yang menjalankan aplikasi Docker pada perangkat Core. Untuk informasi selengkapnya, lihat the section called “Deployment aplikasi Docker”.
- SiteWise Konektor IoT baru yang mengirimkan data perangkat industri dari server OPC-UA ke properti aset di. AWS IoT SiteWise Untuk informasi selengkapnya, lihat the section called “IoT SiteWise”.
- fungsi Lambda yang berjalan tanpa kontainerisasi dapat mengakses sumber daya machine learning dalam grup Greengrass. Untuk informasi selengkapnya, lihat the section called “Mengakses sumber daya machine learning”.
- Support untuk sesi tetap MQTT dengan AWS IoT. Untuk informasi selengkapnya, lihat the section called “Sesi persisten MQTT dengan AWS IoT Core”.
- Lalu lintas MQTT lokal dapat melakukan perjalanan melalui port selain port default 8883. Untuk informasi selengkapnya, lihat the section called “Port MQTT untuk olahpesan lokal”.
- Pilihan `queueFullPolicy` baru di AWS IoT Greengrass Core SDK untuk penerbitan pesan terpercaya dari fungsi Lambda.
- Support untuk menjalankan Node.js 12.x Lambda fungsi pada core.
- Pembaruan O ver-the-air (OTA) dengan integrasi keamanan perangkat keras dapat dikonfigurasi dengan OpenSSL 1.1.
- Peningkatan performa umum dan perbaikan bug.

1.9.4

Perbaikan bug dan peningkatan:

- Peningkatan performa umum dan perbaikan bug.

1.9.3

Fitur-fitur baru:

- Support untuk Armv6l. AWS IoT Greengrass Perangkat lunak Core v1.9.3 atau yang lebih baru dapat diinstal pada distribusi Raspbian dengan arsitektur Armv6l (sebagai contoh, pada perangkat Raspberry Pi Zero).
- OTA membarui pada port 443 dengan ALPN. Greengrass core yang menggunakan port 443 untuk lalu lintas MQTT sekarang mendukung pembaruan perangkat lunak (OTA). over-the-air AWS IoT Greengrassmenggunakan ekstensi Application Layer Protocol Network (ALPN) TLS untuk mengaktifkan koneksi ini. Untuk informasi lebih lanjut, lihat Perbarui OTA AWS IoT Greengrass Perangkat lunak Core dan the section called “Connect pada port 443 atau melalui proksi jaringan”.

Perbaikan bug dan peningkatan:

- Memperbaiki bug yang ditemukan di v1.9.0 yang mencegah fungsi Lambda Python 2.7 mengirimkan muatan biner ke fungsi Lambda lainnya.
- Peningkatan performa umum dan perbaikan bug.

1.9.2

Fitur-fitur baru:

- Support untuk OpenWrt. AWS IoT Greengrass Perangkat lunak inti v1.9.2 atau yang lebih baru dapat diinstal pada OpenWrt distribusi dengan arsitektur Armv8 (AArch64) dan ARMv7L. Saat ini, OpenWrt tidak mendukung inferensi ML.

1.9.1

Perbaikan bug dan peningkatan:

- Memperbaiki bug yang ditemukan di v1.9.0 yang menggagalkan pesan dari `cloud` yang berisi karakter wildcard dalam topik.

1.9.0

Fitur-fitur baru:

- Support untuk Python 3.7 dan Node.js 8.10 Lambda waktu aktif. Fungsi Lambda yang menggunakan Python 3.7 dan Node.js 8.10 waktu aktif sekarang dapat berjalan pada core

AWS IoT Greengrass ini. (AWS IoT Greengrass terus mendukung Python 2.7 dan Node.js 6.10 waktu aktif.)

- Koneksi MQTT yang dioptimalkan. Core Greengrass menetapkan koneksi yang lebih sedikit dengan AWS IoT Core. Perubahan ini dapat mengurangi biaya operasional untuk beban yang didasarkan pada jumlah koneksi.
- Kunci Elliptic Curve (EC) untuk server MQTT lokal. Server MQTT lokal mendukung kunci EC selain kunci RSA. (Sertifikat server MQTT memiliki tanda tangan SHA-256 RSA, terlepas dari jenis kunci.) Untuk informasi selengkapnya, lihat the section called “Keamanan utama”.

Perbaikan bug dan peningkatan:

- Peningkatan performa umum dan perbaikan bug.

1.8.4

Memperbaiki masalah dengan sinkronisasi bayangan dan rekoneksi Certificate Manager perangkat.

Peningkatan performa umum dan perbaikan bug.

1.8.3

Peningkatan performa umum dan perbaikan bug.

1.8.2

Peningkatan performa umum dan perbaikan bug.

1.8.1

Peningkatan performa umum dan perbaikan bug.

1.8.0

Fitur-fitur baru:

- Identitas akses default yang dapat dikonfigurasi untuk fungsi Lambda dalam grup. Pengaturan tingkat grup ini menentukan izin default yang digunakan untuk menjalankan fungsi Lambda. Anda dapat mengatur ID pengguna, ID grup, atau keduanya. Fungsi Lambda individu dapat menimpa identitas akses default grup mereka. Untuk informasi selengkapnya, lihat the section called “Mengatur identitas akses default untuk fungsi Lambda dalam grup”.
- Lalu lintas HTTPS melalui port 443. Komunikasi HTTPS dapat dikonfigurasi untuk perjalanan melalui port 443 daripada port default 8443. Ini melengkapi dukungan AWS

IoT Greengrass untuk ekstensi TLS Application Layer Protocol Network (ALPN) dan mengizinkan semua lalu lintas olahpesan Greengrass—baik MQTT maupun HTTPS—untuk menggunakan port 443. Untuk informasi selengkapnya, lihat the section called “Connect pada port 443 atau melalui proksi jaringan”.

- ID klien yang dapat diprediksi namanya untuk koneksi AWS IoT ini. Perubahan ini mengaktifkan dukungan untuk AWS IoT Device Defender dan AWS IoT Siklus hidup, sehingga Anda dapat menerima notifikasi untuk koneksi, putuskan sambungan, berlangganan, dan berhenti berlangganan acara. Penamaan yang dapat diprediksi juga membuatnya lebih mudah untuk membuat logika di sekitar ID koneksi (sebagai contoh, untuk membuat templat kebijakan langganan berdasarkan atribut sertifikat). Untuk informasi selengkapnya, lihat the section called “ID klien untuk koneksi MQTT dengan AWS IoT”.

Perbaikan bug dan peningkatan:

- Memperbaiki masalah dengan sinkronisasi bayangan dan rekoneksi Certificate Manager perangkat.
- Peningkatan performa umum dan perbaikan bug.

1.7.1

Fitur-fitur baru:

- Konektor Greengrass menyediakan integrasi bawaan dengan infrastruktur lokal, protokol perangkat, AWS, dan layanan cloud lainnya. Untuk informasi selengkapnya, lihat Integrasikan dengan layanan dan protokol menggunakan konektor.
- AWS IoT Greengrass memperluas AWS Secrets Manager ke perangkat core, yang membuat kata sandi, token, dan rahasia lainnya tersedia untuk konektor dan fungsi Lambda. Rahasia dienkripsi saat transit dan saat tidak digunakan. Untuk informasi selengkapnya, lihat Men-deploy rahasia ke core.
- Support untuk root perangkat keras dari opsi keamanan kepercayaan. Untuk informasi selengkapnya, lihat the section called “Integrasi keamanan perangkat keras”.
- Isolasi dan izin pengaturan yang memungkinkan fungsi Lambda untuk menjalankan tanpa kontainer Greengrass dan menggunakan izin dari pengguna tertentu dan grup. Untuk informasi selengkapnya, lihat the section called “Mengontrol eksekusi fungsi Greengrass Lambda”.
- Anda dapat menjalankan AWS IoT Greengrass dalam kontainer Docker (pada Windows, MacOS, atau Linux) dengan mengonfigurasi grup Greengrass Anda agar berjalan tanpa kontainerisasi. Untuk informasi selengkapnya, lihat the section called “Jalankan AWS IoT Greengrass di kontainer Docker”.

- Olahpesan MQTT pada port 443 dengan Application Layer Protocol Negotiation (ALPN) atau koneksi melalui proxy jaringan. Untuk informasi selengkapnya, lihat the section called “Connect pada port 443 atau melalui proksi jaringan”.
- Runtime pembelajaran mendalam SageMaker Neo, yang mendukung model pembelajaran mesin yang telah dioptimalkan oleh kompiler pembelajaran mendalam SageMaker Neo. Untuk informasi tentang waktu aktif deep learning Neo, lihat the section called “Waktu aktif dan perpustakaan untuk inferensi ML”.
- Support untuk Raspbian Stretch (2018-06-27) pada perangkat core Raspberry Pi.

Perbaikan bug dan peningkatan:

- Peningkatan performa umum dan perbaikan bug.

Selain itu, fitur berikut tersedia pada rilis ini:

- Penguji Perangkat AWS IoT pada AWS IoT Greengrass, yang dapat Anda gunakan untuk memverifikasi bahwa arsitektur CPU, konfigurasi kernel, dan driver bekerja dengan AWS IoT Greengrass. Untuk informasi selengkapnya, lihat Menggunakan AWS IoT Penguji Perangkat untuk AWS IoT Greengrass V1.
- Paket perangkat lunak AWS IoT Greengrass Core, AWS IoT Greengrass Core SDK, dan AWS IoT Greengrass Machine Learning SDK tersedia untuk diunduh melalui Amazon. CloudFront Untuk informasi selengkapnya, lihat the section called “AWS IoT Greengrass unduh”.

1.6.1

Fitur-fitur baru:

- Lambda yang dapat dieksekusi yang menjalankan kode biner pada core Greengrass. Gunakan AWS IoT Greengrass Core SDK for C baru untuk menulis Lambda yang dapat dieksekusi di C dan C ++. Untuk informasi selengkapnya, lihat the section called “Lambda yang dapat dieksekusi”.
- Cache pesan penyimpanan lokal opsional yang dapat bertahan di restart. Anda dapat mengonfigurasi pengaturan penyimpanan untuk pesan MQTT yang antre untuk pemrosesan. Untuk informasi selengkapnya, lihat the section called “Antrean pesan MQTT”.
- Interval coba ulang maksimum yang dapat dikonfigurasi pada saat perangkat core terputus. Untuk informasi lebih lanjut, lihat properti `mqttMaxConnectionRetryInterval` di the section called “AWS IoT Greengrass file konfigurasi core”.
- Akses sumber daya lokal ke direktori host /proc. Untuk informasi selengkapnya, lihat Akses sumber daya lokal.

- Direktori tulis yang dapat dikonfigurasi. Perangkat lunak core AWS IoT Greengrass dapat digunakan untuk hanya-baca dan baca-tulis lokasi. Untuk informasi selengkapnya, lihat the section called “Mengonfigurasi direktori tulis”.

Perbaikan bug dan peningkatan:

- Peningkatan performa untuk menerbitkan pesan di core Greengrass dan antara perangkat dan core.
- Mengurangi sumber daya komputasi yang diperlukan untuk memproses catatan yang dihasilkan oleh fungsi Lambda yang ditetapkan pengguna.

1.5.0

Fitur-fitur baru:

- AWS IoT Greengrass Machine Learning (ML) Inferensi umumnya tersedia. Anda dapat melakukan inferensi ML lokal pada perangkat AWS IoT Greengrass menggunakan model yang dibuat dan dilatih di cloud. Untuk informasi selengkapnya, lihat Lakukan inferensi machine learning.
- Sekarang, Fungsi Greengrass Lambda mendukung data biner sebagai input muatan, selain JSON. Untuk menggunakan fitur ini, Anda harus memperbarui ke Core SDK AWS IoT Greengrass versi 1.1.0, yang dapat Anda unduh dari halaman mengunduh AWS IoT Greengrass Core SDK ini.

Perbaikan bug dan peningkatan:

- Mengurangi jejak memori secara keseluruhan.
- Peningkatan performa untuk mengirim pesan ke cloud.
- Peningkatan performa dan stabilitas untuk agen mengunduh, Certificate Manager perangkat, dan agen pembaruan OTA.
- Perbaikan bug minor.

1.3.0

Fitur-fitur baru:

- Agen pembaruan O ver-the-air (OTA) yang mampu menangani pekerjaan pembaruan Greengrass yang diterapkan di cloud. Agen ini ditemukan di bawah direktori `/greengrass/ota` baru. Untuk informasi selengkapnya, lihat Perbarui OTA AWS IoT Greengrass perangkat lunak Core.

- Fitur akses sumber daya lokal memungkinkan fungsi Greengrass Lambda untuk mengakses sumber daya lokal, seperti perangkat periferal dan volume. Untuk informasi selengkapnya, lihat Akses sumber daya lokal dengan fungsi dan konektor Lambda.

1.1.0

Fitur-fitur baru:

- Men-deploy grup AWS IoT Greengrass dapat diatur ulang dengan menghapus fungsi Lambda, langganan, dan konfigurasi. Untuk informasi selengkapnya, lihat the section called "Atur ulang deployment".
- Support untuk Node.js 6.10 dan Java 8 Lambda waktu aktif, selain Python 2.7.

Untuk bermigrasi dari versi core AWS IoT Greengrass sebelumnya:

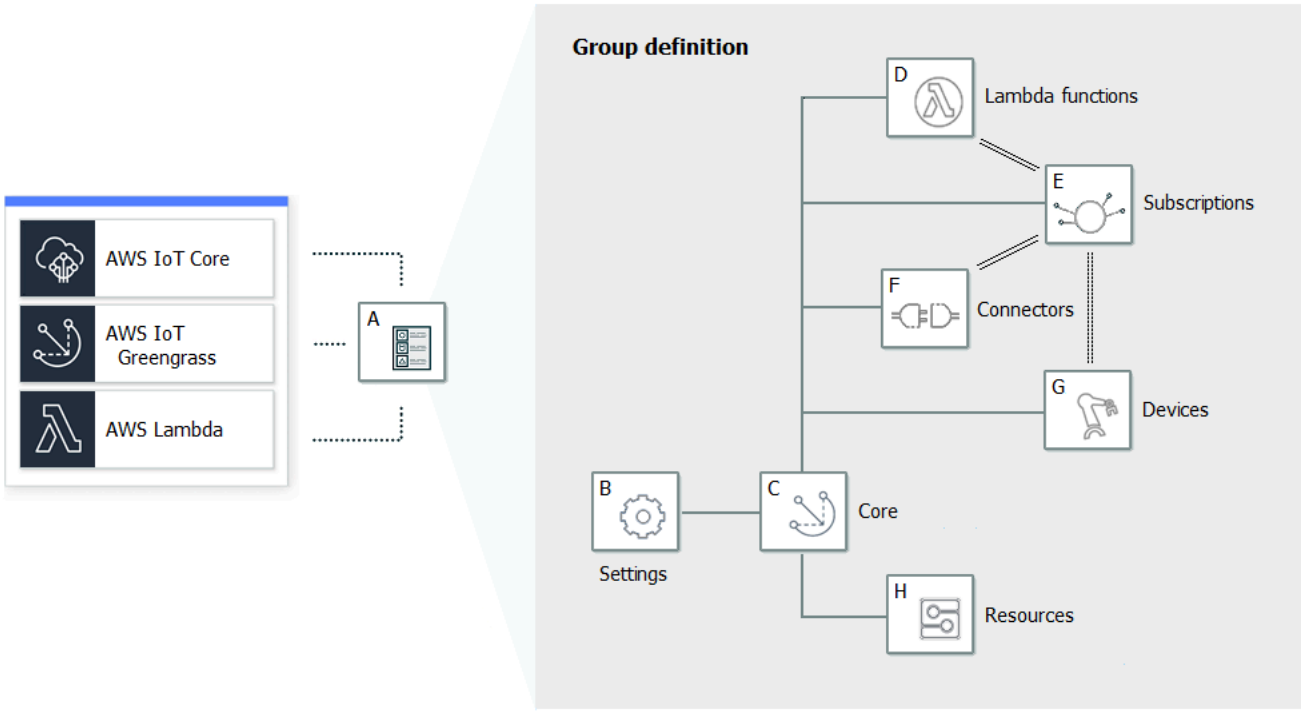
- Salin sertifikat dari folder `/greengrass/configuration/certs` ke `/greengrass/certs`.
- Salin `/greengrass/configuration/config.json` ke `/greengrass/config/config.json`.
- Jalankan `/greengrass/ggc/core/greengrassd` sebagai ganti `/greengrass/greengrassd`.
- Men-deploy grup ke core baru.

1.0.0

Versi awal

AWS IoT Greengrass Grup

Sebuah grup Greengrass adalah kumpulan pengaturan dan komponen, seperti core Greengrass, perangkat, dan langganan. Grup yang digunakan untuk mendefinisikan lingkup interaksi. Sebagai contoh, grup mungkin mewakili satu lantai bangunan, satu truk, atau seluruh lokasi pertambangan. Diagram berikut menunjukkan komponen yang dapat membentuk grup Greengrass.



Pada diagram sebelumnya:

A: Definisi grup Greengrass

Informasi tentang pengaturan grup dan komponen.

B: Pengaturan grup Greengrass

Ini termasuk:

- Peran grup Greengrass.
- Otoritas sertifikat dan konfigurasi koneksi lokal.
- Informasi konektivitas core Greengrass.
- Lingkungan waktu aktif Lambda default. Untuk informasi selengkapnya, lihat the section called “Pengaturan kontainerisasi default untuk fungsi Lambda dalam grup”.
- CloudWatch dan konfigurasi log lokal. Untuk informasi selengkapnya, lihat the section called “Pemantauan dengan AWS IoT Greengrass log”.

C: Core Greengrass

Hal (perangkat) AWS IoT yang mewakili core Greengrass. Untuk informasi selengkapnya, lihat the section called “Mengonfigurasi AWS IoT Greengrass core”.

D: Definisi fungsi Lambda

Daftar fungsi Lambda yang berjalan secara lokal pada core, dengan data konfigurasi terkait. Untuk informasi selengkapnya, lihat Jalankan fungsi Lambda lokal.

E: Definisi langganan

Daftar langganan yang memungkinkan komunikasi menggunakan pesan MQTT. Sebuah langganan mendefinisikan:

- Sumber pesan dan target pesan. Ini bisa berupa perangkat klien, fungsi Lambda, konektorAWS IoT Core, dan layanan bayangan lokal.
- Topik atau subjek yang digunakan untuk memfilter pesan.

Untuk informasi selengkapnya, lihat the section called “Langganan yang dikelola di dalam alur kerja pesan MQTT”.

F: Definisi konektor

Daftar konektor yang berjalan secara lokal pada core, dengan data konfigurasi terkait. Untuk informasi selengkapnya, lihat Integrasikan dengan layanan dan protokol menggunakan konektor.

G: Definisi perangkat

Daftar AWS IoT hal-hal (dikenal sebagai perangkat klien atau perangkat) yang merupakan anggota grup Greengrass, dengan data konfigurasi terkait. Untuk informasi selengkapnya, lihat the section called “Perangkat di AWS IoT Greengrass”.

H: Definisi sumber daya

Daftar sumber daya lokal, sumber daya machine learning, dan sumber daya rahasia pada core Greengrass, dengan data konfigurasi terkait. Untuk informasi lebih lanjut, lihat Akses sumber daya lokal, Tampilkan inferensi machine learning, dan Men-deploy rahasia ke core.

Ketika di-deploy, definisi grup Greengrass, fungsi Lambda, konektor, sumber daya, dan tabel berlangganan disalin ke perangkat core. Untuk informasi selengkapnya, lihat Men-deploy AWS IoT Greengrass grup.

Perangkat di AWS IoT Greengrass

Sebuah grup Greengrass dapat berisi dua jenis perangkat AWS IoT ini:

Core Greengrass

Core Greengrass adalah perangkat yang menjalankan perangkat lunak core AWS IoT Greengrass tersebut, yang memungkinkannya berkomunikasi langsung dengan AWS IoT Core dan Layanan AWS IoT Greengrass ini. Sebuah core memiliki sertifikat perangkat sendiri yang digunakan untuk autentikasi dengan AWS IoT Core. Ini memiliki bayangan perangkat dan entri di registri AWS IoT Core ini. Greengrass core menjalankan runtime Lambda lokal, agen penyebaran, dan pelacak alamat IP yang mengirimkan informasi alamat IP ke layanan untuk memungkinkan perangkat klien menemukan informasi koneksi grup AWS IoT Greengrass dan inti mereka secara otomatis. Untuk informasi selengkapnya, lihat the section called “Mengonfigurasi AWS IoT Greengrass core”.

> **Note**
>
> Sebuah grup Greengrass harus berisi hanya satu core.

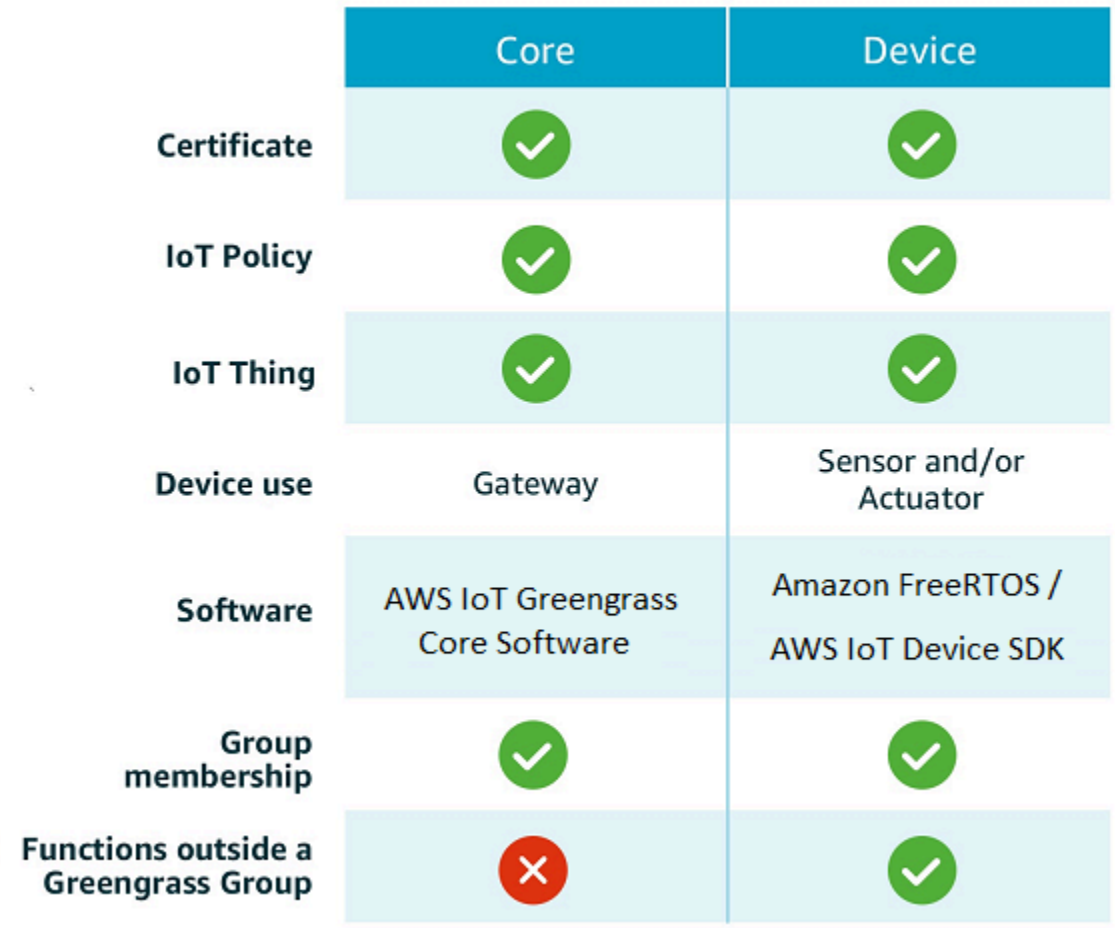
Perangkat klien

Perangkat klien (juga disebut perangkat yang terhubung, perangkat Greengrass, atau perangkat) adalah perangkat yang terhubung ke inti Greengrass melalui MQTT. Mereka memiliki sertifikat perangkat mereka sendiri untuk AWS IoT Core otentikasi, bayangan perangkat, dan entri dalam AWS IoT Core registri. Perangkat klien dapat menjalankan FreeRTOS atau menggunakan AWS IoTDevice SDK AWS IoT Greengrassatau Discovery API untuk mendapatkan informasi penemuan yang digunakan untuk menghubungkan dan mengautentikasi dengan inti dalam grup Greengrass yang sama. Untuk mempelajari cara menggunakan AWS IoT konsol untuk membuat dan mengonfigurasi perangkat klienAWS IoT Greengrass, lihatthe section called “Modul 4: Berinteraksilah dengan perangkat klien dalamAWS IoT Greengrasskelompok”. Atau, untuk contoh yang menunjukkan cara menggunakan perangkat AWS CLI untuk membuat dan mengonfigurasi perangkat klienAWS IoT Greengrass, lihat create-device-definitiondi Referensi AWS CLI Perintah.

Dalam grup Greengrass, Anda dapat membuat langganan yang memungkinkan perangkat klien berkomunikasi melalui MQTT dengan fungsi Lambda, konektor, dan perangkat klien lainnya dalam grup, dan dengan atau layanan bayangan lokal. AWS IoT Core Pesan MQTT diarahkan melalui core. Jika perangkat inti kehilangan konektivitas ke cloud, perangkat klien dapat terus berkomunikasi melalui jaringan lokal. Perangkat klien dapat bervariasi dalam ukuran, dari perangkat berbasis mikrokontroler yang lebih kecil hingga peralatan besar. Saat ini, grup Greengrass dapat berisi hingga 2.500 perangkat klien. Perangkat klien dapat menjadi anggota hingga 10 grup.

> **Note**
>
> OPC-UA adalah standar pertukaran informasi untuk komunikasi industri. Untuk mengimplementasikan dukungan OPC-UA pada inti Greengrass, Anda dapat menggunakan konektor IoT. SiteWise Konektor mengirimkan data perangkat industri dari server OPC-UA ke properti aset di AWS IoT SiteWise.

Tabel berikut menunjukkan bagaimana jenis perangkat ini terkait.



Perangkat lunak Core AWS IoT Greengrass menyimpan sertifikat di dua lokasi:

- Sertifikat perangkat Core di `/greengrass-root/certs`. Biasanya, sertifikat perangkat Core bernama *`hash`*`.cert.pem` (sebagai contoh, `86c84488a5.cert.pem`). Sertifikat ini digunakan oleh klien AWS IoT untuk saling mengautentikasi ketika core menghubungkan ke AWS IoT Core dan layanan AWS IoT Greengrass ini.

- Sertifikat server MTT di */greengrass-root*/ggc/var/state/server. Sertifikat server MQTT bernama server.crt. Sertifikat ini digunakan untuk saling mengautentikasi antara server MQTT lokal (pada Core Greengrass) dan perangkat Greengrass.

> **Note**
>
> *Greengrass-root* mewakili jalur di mana perangkat lunak Core AWS IoT Greengrass diinstal pada perangkat Anda. Biasanya, adalah direktori /greengrass ini.

SDK

SDK AWS-disediakan berikut digunakan untuk bekerja dengan AWS IoT Greengrass:

AWS SDK

Gunakan AWS SDK untuk membangun aplikasi yang berinteraksi dengan layanan AWS apapun, termasuk Amazon S3, Amazon DynamoDB, AWS IoT, AWS IoT Greengrass, dan banyak lagi. Dalam konteks AWS IoT Greengrass, Anda dapat menggunakan AWS SDK dalam fungsi Lambda di-deploy untuk melakukan panggilan langsung ke layanan AWS ini. Untuk informasi selengkapnya, lihat AWS SDK.

> **Note**
>
> Operasi khusus untuk Greengrass yang tersedia di AWS SDK juga tersedia di AWS IoT Greengrass API dan AWS CLI.

AWS IoT Perangkat SDK

Ini AWS IoT Perangkat SDK yang membantu perangkat terhubung ke AWS IoT Core dan AWS IoT Greengrass. Untuk informasi lebih lanjut, lihat AWS IoT Perangkat SDK di Panduan DeveloperAWS IoT.

Perangkat klien dapat menggunakan salah satu platform AWS IoT Device SDK v2 untuk menemukan informasi konektivitas untuk inti Greengrass. Informasi konektivitas meliputi:

- ID grup Greengrass yang dimiliki perangkat klien.
- Alamat IP dari core Greengrass di setiap grup. Ini juga disebut titik akhir core.

- Sertifikat CA grup, di mana perangkat menggunakannya untuk saling mengautentikasi dengan core. Untuk informasi selengkapnya, lihat the section called “Alur kerja koneksi perangkat”.

> **Note**
>
> Pada v1 dari AWS IoT Perangkat SDK, hanya platform C++ dan Python yang menyediakan dukungan penemuan bawaan.

AWS IoT Greengrass Core SDK

Ini AWS IoT Greengrass Core SDK yang memungkinkan fungsi Lambda untuk berinteraksi dengan core Greengrass, mempublikasikan pesan ke AWS IoT, berinteraksi dengan layanan bayangan lokal, memanggil fungsi Lambda di-deploy lainnya, dan mengakses sumber daya rahasia. SDK ini digunakan oleh fungsi Lambda yang berjalan pada AWS IoT Greengrass core. Untuk informasi selengkapnya, lihat AWS IoT GreengrassSDK inti.

AWS IoT Greengrass Machine Learning SDK

Ini AWS IoT Greengrass Machine Learning SDK yang memungkinkan fungsi Lambda untuk mengonsumsi model machine learning yang di-deploy ke core Greengrass sebagai sumber daya machine learning. SDK ini digunakan oleh fungsi Lambda yang berjalan pada core AWS IoT Greengrass dan berinteraksi dengan layanan inferensi lokal. Untuk informasi selengkapnya, lihat AWS IoT GreengrassSDK Machine Learning.

Platform dan persyaratan yang didukung

Tab berikut mencantumkan platform dan persyaratan yang didukung untuk perangkat lunak core AWS IoT Greengrass ini.

> **Note**
>
> Anda dapat mengunduh AWS IoT Greengrass perangkat lunak Core dari AWS IoT Greengrass unduhan perangkat lunak Core.

GGC v1.11

Platform yang didukung:

- Arsitektur: Armv7l
  - OS: Linux
  - OS: Linux (OpenWrt)
- Arsitektur: Armv8 (AArch64)
  - OS: Linux
  - OS: Linux (OpenWrt)
- arsitektur: Armv6l
  - OS: Linux
- Arsitektur: x86_64
  - OS: Linux
- Platform Windows, MacOS, dan Linux dapat menjalankan AWS IoT Greengrass dalam kontainer Docker. Untuk informasi selengkapnya, lihat the section called “Jalankan AWS IoT Greengrass di kontainer Docker”.

Persyaratan:

- Minimum 128 MB ruang disk yang tersedia untuk perangkat lunak Core AWS IoT Greengrass ini. Jika Anda menggunakan Agen pembaruan OTA, minimumnya adalah 400 MB.
- Minimum 128 MB RAM dialokasikan untuk perangkat lunak Core AWS IoT Greengrass ini. Dengan mengaktifkan Pengelola aliran ini, minimumnya adalah 198 MB RAM.

  > **Note**
  >
  > Pengelola aliran diaktifkan secara default jika Anda menggunakan opsi Pembuatan grup default pada konsol AWS IoT untuk membuat grup Greengrass Anda.

- Versi kernel Linux:
  - Kernel Linux versi 4.4 atau yang lebih baru diperlukan untuk mendukung menjalankan AWS IoT Greengrass dengan Kontainer.
  - Kernel Linux versi 3.17 atau yang lebih baru diperlukan untuk mendukung menjalankan AWS IoT Greengrass tanpa kontainer. Dalam konfigurasi ini, fungsi Lambda kontainerisasi default untuk grup Greengrass harus diatur ke Tanpa kontainer. Untuk petunjuk, lihat the section called “Pengaturan kontainerisasi default untuk fungsi Lambda dalam grup”.

- GNU C Library (glibc) versi 2.14 atau yang lebih baru. OpenWrt distribusi memerlukan musl C Library versi 1.1.16 atau yang lebih baru.
- Direktori `/var/run` harus ada pada perangkat.
- File `/dev/stdin`, `/dev/stdout`, dan `/dev/stderr` harus tersedia.
- Perlindungan hardlink dan softlink harus diaktifkan pada perangkat. Jika tidak, AWS IoT Greengrass hanya dapat dijalankan dalam mode tidak aman, menggunakan bendera `-i` ini.
- Konfigurasi kernel Linux berikut harus diaktifkan pada perangkat:
  - Namespace:
    - CONFIG_IPC_NS
    - CONFIG_UTS_NS
    - CONFIG_USER_NS
    - CONFIG_PID_NS
  - Cgroups:
    - CONFIG_CGROUP_DEVICE
    - CONFIG_CGROUPS
    - CONFIG_MEMCG

    Kernel harus mendukung cgroups. Persyaratan berikut berlaku saat menjalankan AWS IoT Greengrass dengan kontainer:
    - Cgroup memori harus diaktifkan dan dipasang untuk mengizinkan AWS IoT Greengrass mengatur batas memori pada fungsi Lambda.
    - Cgroup perangkat harus diaktifkan dan dipasang jika fungsi Lambda dengan akses sumber daya lokal digunakan untuk membuka file pada perangkat Core AWS IoT Greengrass ini.
  - Lainnya:
    - CONFIG_POSIX_MQUEUE
    - CONFIG_OVERLAY_FS
    - CONFIG_HAVE_ARCH_SECCOMP_FILTER
    - CONFIG_SECCOMP_FILTER
    - CONFIG_KEYS
    - CONFIG_SECCOMP
    - CONFIG_SHMEM
- Sertifikat root untuk Amazon S3 dan AWS IoT harus ada di penyimpanan kepercayaan sistem.

- Pengelola aliran membutuhkan Java 8 waktu aktif dan minimal 70 MB RAM selain basis kebutuhan memori perangkat lunak Core AWS IoT Greengrass ini. Pengelola aliran diaktifkan secara default ketika Anda menggunakan opsi Pembuatan grup default pada konsol AWS IoT tersebut. Manajer aliran tidak didukung pada OpenWrt distribusi.
- Perpustakaan yang mendukung AWS Lambda waktu aktif diperlukan oleh fungsi Lambda yang Anda inginkan untuk berjalan secara lokal. Perpustakaan yang diperlukan harus diinstal pada core dan ditambahkan ke variabel lingkungan `PATH` ini. Beberapa perpustakaan dapat diinstal pada Core yang sama.
  - Python versi 3.8 untuk fungsi yang menggunakan waktu aktif Python 3.8.
  - Python versi 3.7 untuk fungsi yang menggunakan waktu aktif Python 3.7.
  - Python versi 2.7 untuk fungsi yang menggunakan waktu aktif Python 2.7.
  - Node.js versi 12.x untuk fungsi yang menggunakan Node.js 12.x waktu aktif.
  - Java versi 8 atau yang lebih baru untuk fungsi yang menggunakan Java 8 waktu aktif.

    > **Note**
    >
    > Menjalankan Java pada OpenWrt distribusi tidak didukung secara resmi. Namun, jika OpenWrt build Anda memiliki dukungan Java, Anda mungkin dapat menjalankan fungsi Lambda yang ditulis di Java di perangkat Anda. OpenWrt

    Untuk informasi lebih lanjut tentang dukungan AWS IoT Greengrass untuk waktu aktif Lambda, lihat Jalankan fungsi Lambda lokal.
- Perintah shell berikut (bukan BusyBox variannya) diperlukan oleh agen pembaruan over-the-air (OTA):
  - `wget`
  - `realpath`
  - `tar`
  - `readlink`
  - `basename`
  - `dirname`
  - `pidof`
  - `df`
  - `grep`

- `umount`
- `mv`
- `gzip`
- `mkdir`
- `rm`
- `ln`
- `cut`
- `cat`
- `/bin/bash`

GGC v1.10

Platform yang didukung:

- Arsitektur: Armv7l
  - OS: Linux
  - OS: Linux (OpenWrt)
- Arsitektur: Armv8 (AArch64)
  - OS: Linux
  - OS: Linux (OpenWrt)
- arsitektur: Armv6l
  - OS: Linux
- Arsitektur: x86_64
  - OS: Linux
- Platform Windows, MacOS, dan Linux dapat menjalankan AWS IoT Greengrass dalam kontainer Docker. Untuk informasi selengkapnya, lihat the section called “Jalankan AWS IoT Greengrass di kontainer Docker”.

Persyaratan:

- Minimum 128 MB ruang disk yang tersedia untuk perangkat lunak Core AWS IoT Greengrass ini. Jika Anda menggunakan Agen pembaruan OTA, minimumnya adalah 400 MB.

- Minimum 128 MB RAM dialokasikan untuk perangkat lunak Core AWS IoT Greengrass ini. Dengan mengaktifkan Pengelola aliran ini, minimumnya adalah 198 MB RAM.

  > **Note**
  >
  > Pengelola aliran diaktifkan secara default jika Anda menggunakan opsi Pembuatan grup default pada konsol AWS IoT untuk membuat grup Greengrass Anda.

- Versi kernel Linux:
  - Kernel Linux versi 4.4 atau yang lebih baru diperlukan untuk mendukung menjalankan AWS IoT Greengrass dengan Kontainer.
  - Kernel Linux versi 3.17 atau yang lebih baru diperlukan untuk mendukung menjalankan AWS IoT Greengrass tanpa kontainer. Dalam konfigurasi ini, fungsi Lambda kontainerisasi default untuk grup Greengrass harus diatur ke Tanpa kontainer. Untuk petunjuk, lihat the section called “Pengaturan kontainerisasi default untuk fungsi Lambda dalam grup”.
- GNU C Library (glibc) versi 2.14 atau yang lebih baru. OpenWrt distribusi memerlukan musl C Library versi 1.1.16 atau yang lebih baru.
- Direktori `/var/run` harus ada pada perangkat.
- File `/dev/stdin`, `/dev/stdout`, dan `/dev/stderr` harus tersedia.
- Perlindungan hardlink dan softlink harus diaktifkan pada perangkat. Jika tidak, AWS IoT Greengrass hanya dapat dijalankan dalam mode tidak aman, menggunakan bendera `-i` ini.
- Konfigurasi kernel Linux berikut harus diaktifkan pada perangkat:
  - Namespace:
    - CONFIG_IPC_NS
    - CONFIG_UTS_NS
    - CONFIG_USER_NS
    - CONFIG_PID_NS
  - Cgroups:
    - CONFIG_CGROUP_DEVICE
    - CONFIG_CGROUPS
    - CONFIG_MEMCG

    Kernel harus mendukung cgroups. Persyaratan berikut berlaku saat menjalankan AWS IoT Greengrass dengan kontainer:

    - Cgroup memori harus diaktifkan dan dipasang untuk mengizinkan AWS IoT Greengrass mengatur batas memori pada fungsi Lambda.
    - Cgroup perangkat harus diaktifkan dan dipasang jika fungsi Lambda dengan akses sumber daya lokal digunakan untuk membuka file pada perangkat Core AWS IoT Greengrass ini.
  - Lainnya:
    - CONFIG_POSIX_MQUEUE
    - CONFIG_OVERLAY_FS
    - CONFIG_HAVE_ARCH_SECCOMP_FILTER
    - CONFIG_SECCOMP_FILTER
    - CONFIG_KEYS
    - CONFIG_SECCOMP
    - CONFIG_SHMEM
- Sertifikat root untuk Amazon S3 dan AWS IoT harus ada di penyimpanan kepercayaan sistem.
- Pengelola aliran membutuhkan Java 8 waktu aktif dan minimal 70 MB RAM selain basis kebutuhan memori perangkat lunak Core AWS IoT Greengrass ini. Pengelola aliran diaktifkan secara default ketika Anda menggunakan opsi Pembuatan grup default pada konsol AWS IoT tersebut. Manajer aliran tidak didukung pada OpenWrt distribusi.
- Perpustakaan yang mendukung AWS Lambda waktu aktif diperlukan oleh fungsi Lambda yang Anda inginkan untuk berjalan secara lokal. Perpustakaan yang diperlukan harus diinstal pada core dan ditambahkan ke variabel lingkungan `PATH` ini. Beberapa perpustakaan dapat diinstal pada Core yang sama.
  - Python versi 3.7 untuk fungsi yang menggunakan waktu aktif Python 3.7.
  - Python versi 2.7 untuk fungsi yang menggunakan waktu aktif Python 2.7.
  - Node.js versi 12.x untuk fungsi yang menggunakan Node.js 12.x waktu aktif.
  - Java versi 8 atau yang lebih baru untuk fungsi yang menggunakan Java 8 waktu aktif.

    > **Note**
    >
    > Menjalankan Java pada OpenWrt distribusi tidak didukung secara resmi. Namun, jika OpenWrt build Anda memiliki dukungan Java, Anda mungkin dapat menjalankan fungsi Lambda yang ditulis di Java di perangkat Anda. OpenWrt

Untuk informasi lebih lanjut tentang dukungan AWS IoT Greengrass untuk waktu aktif Lambda, lihat Jalankan fungsi Lambda lokal.

- Perintah shell berikut (bukan BusyBox variannya) diperlukan oleh agen pembaruan over-the-air (OTA):
  - `wget`
  - `realpath`
  - `tar`
  - `readlink`
  - `basename`
  - `dirname`
  - `pidof`
  - `df`
  - `grep`
  - `umount`
  - `mv`
  - `gzip`
  - `mkdir`
  - `rm`
  - `ln`
  - `cut`
  - `cat`
  - `/bin/bash`

GGC v1.9

Platform yang didukung:

- Arsitektur: Armv7l
  - OS: Linux
  - OS: Linux (OpenWrt)
- Arsitektur: Armv8 (AArch64)

    - OS: Linux
    - OS: Linux (OpenWrt)
- arsitektur: Armv6l
    - OS: Linux
- Arsitektur: x86_64
    - OS: Linux
- Platform Windows, MacOS, dan Linux dapat menjalankan AWS IoT Greengrass dalam kontainer Docker. Untuk informasi selengkapnya, lihat the section called “Jalankan AWS IoT Greengrass di kontainer Docker”.

Persyaratan:

- Minimum 128 MB ruang disk yang tersedia untuk perangkat lunak Core AWS IoT Greengrass ini. Jika Anda menggunakan Agen pembaruan OTA, minimumnya adalah 400 MB.
- Minimum 128 MB RAM dialokasikan untuk perangkat lunak Core AWS IoT Greengrass ini.
- Versi kernel Linux:
    - Kernel Linux versi 4.4 atau yang lebih baru diperlukan untuk mendukung menjalankan AWS IoT Greengrass dengan Kontainer.
    - Kernel Linux versi 3.17 atau yang lebih baru diperlukan untuk mendukung menjalankan AWS IoT Greengrass tanpa kontainer. Dalam konfigurasi ini, fungsi Lambda kontainerisasi default untuk grup Greengrass harus diatur ke Tanpa kontainer. Untuk petunjuk, lihat the section called “Pengaturan kontainerisasi default untuk fungsi Lambda dalam grup”.
- GNU C Library (glibc) versi 2.14 atau yang lebih baru. OpenWrt distribusi memerlukan musl C Library versi 1.1.16 atau yang lebih baru.
- Direktori `/var/run` harus ada pada perangkat.
- File `/dev/stdin`, `/dev/stdout`, dan `/dev/stderr` harus tersedia.
- Perlindungan hardlink dan softlink harus diaktifkan pada perangkat. Jika tidak, AWS IoT Greengrass hanya dapat dijalankan dalam mode tidak aman, menggunakan bendera `-i` ini.
- Konfigurasi kernel Linux berikut harus diaktifkan pada perangkat:
    - Namespace:
        - CONFIG_IPC_NS
        - CONFIG_UTS_NS

    - CONFIG_USER_NS
    - CONFIG_PID_NS
  - Cgroups:
    - CONFIG_CGROUP_DEVICE
    - CONFIG_CGROUPS
    - CONFIG_MEMCG

    Kernel harus mendukung cgroups. Persyaratan berikut berlaku saat menjalankan AWS IoT Greengrass dengan kontainer:

    - Cgroup memori harus diaktifkan dan dipasang untuk mengizinkan AWS IoT Greengrass mengatur batas memori pada fungsi Lambda.
    - Cgroup perangkat harus diaktifkan dan dipasang jika fungsi Lambda dengan akses sumber daya lokal digunakan untuk membuka file pada perangkat Core AWS IoT Greengrass ini.
  - Lainnya:
    - CONFIG_POSIX_MQUEUE
    - CONFIG_OVERLAY_FS
    - CONFIG_HAVE_ARCH_SECCOMP_FILTER
    - CONFIG_SECCOMP_FILTER
    - CONFIG_KEYS
    - CONFIG_SECCOMP
    - CONFIG_SHMEM
- Sertifikat root untuk Amazon S3 dan AWS IoT harus ada di penyimpanan kepercayaan sistem.
- Perpustakaan yang mendukung AWS Lambda waktu aktif diperlukan oleh fungsi Lambda yang Anda inginkan untuk berjalan secara lokal. Perpustakaan yang diperlukan harus diinstal pada core dan ditambahkan ke variabel lingkungan `PATH` ini. Beberapa perpustakaan dapat diinstal pada Core yang sama.
  - Python versi 2.7 untuk fungsi yang menggunakan waktu aktif Python 2.7.
  - Python versi 3.7 untuk fungsi yang menggunakan waktu aktif Python 3.7.
  - Node.js versi 6.10 atau yang lebih baru untuk fungsi yang menggunakan waktu aktif Node.js 6.10.
  - Node.js versi 8.10 atau yang lebih baru untuk fungsi yang menggunakan waktu aktif Node.js 8.10.

- Java versi 8 atau yang lebih baru untuk fungsi yang menggunakan Java 8 waktu aktif.

  > **Note**
  >
  > Menjalankan Java pada OpenWrt distribusi tidak didukung secara resmi. Namun, jika OpenWrt build Anda memiliki dukungan Java, Anda mungkin dapat menjalankan fungsi Lambda yang ditulis di Java di perangkat Anda. OpenWrt

  Untuk informasi lebih lanjut tentang dukungan AWS IoT Greengrass untuk waktu aktif Lambda, lihat Jalankan fungsi Lambda lokal.

- Perintah shell berikut (bukan BusyBox variannya) diperlukan oleh agen pembaruan over-the-air (OTA):
  - `wget`
  - `realpath`
  - `tar`
  - `readlink`
  - `basename`
  - `dirname`
  - `pidof`
  - `df`
  - `grep`
  - `umount`
  - `mv`
  - `gzip`
  - `mkdir`
  - `rm`
  - `ln`
  - `cut`
  - `cat`

GGC v1.8

- Platform yang didukung:
  - Arsitektur: ARMv7L; OS: Linux
  - Arsitektur: x86_64; OS: Linux
  - Arsitektur: Armv8 (AArch64); OS: Linux
  - Platform Windows, MacOS, dan Linux dapat menjalankan AWS IoT Greengrass dalam kontainer Docker. Untuk informasi selengkapnya, lihat the section called “Jalankan AWS IoT Greengrass di kontainer Docker”.
  - Platform Linux dapat menjalankan versi AWS IoT Greengrass dengan fungsi terbatas menggunakan snap Greengrass, yang tersedia melalui Snapcraft. Untuk informasi selengkapnya, lihat the section called “AWS IoT Greengrass perangkat lunak snap”.
- Item berikut diperlukan:
  - Minimum 128 MB ruang disk yang tersedia untuk perangkat lunak Core AWS IoT Greengrass ini. Jika Anda menggunakan Agen pembaruan OTA, minimumnya adalah 400 MB.
  - Minimum 128 MB RAM dialokasikan untuk perangkat lunak Core AWS IoT Greengrass ini.
  - Versi kernel Linux:
    - Kernel Linux versi 4.4 atau yang lebih baru diperlukan untuk mendukung menjalankan AWS IoT Greengrass dengan Kontainer.
    - Kernel Linux versi 3.17 atau yang lebih baru diperlukan untuk mendukung menjalankan AWS IoT Greengrass tanpa kontainer. Dalam konfigurasi ini, fungsi Lambda kontainerisasi default untuk grup Greengrass harus diatur ke Tanpa kontainer. Untuk petunjuk, lihat the section called “Pengaturan kontainerisasi default untuk fungsi Lambda dalam grup”.
  - GNU C Library (glibc) versi 2.14 atau yang lebih baru.
  - Direktori `/var/run` harus ada pada perangkat.
  - File `/dev/stdin`, `/dev/stdout`, dan `/dev/stderr` harus tersedia.
  - Perlindungan hardlink dan softlink harus diaktifkan pada perangkat. Jika tidak, AWS IoT Greengrass hanya dapat dijalankan dalam mode tidak aman, menggunakan bendera `-i` ini.
  - Konfigurasi kernel Linux berikut harus diaktifkan pada perangkat:
    - Namespace:
      - CONFIG_IPC_NS
      - CONFIG_UTS_NS
      - CONFIG_USER_NS

    - CONFIG_PID_NS
  - Cgroups:
    - CONFIG_CGROUP_DEVICE
    - CONFIG_CGROUPS
    - CONFIG_MEMCG

    Kernel harus mendukung cgroups. Persyaratan berikut berlaku saat menjalankan AWS IoT Greengrass dengan kontainer:

    - Cgroup memori harus diaktifkan dan dipasang untuk mengizinkan AWS IoT Greengrass mengatur batas memori pada fungsi Lambda.
    - Cgroup perangkat harus diaktifkan dan dipasang jika fungsi Lambda dengan akses sumber daya lokal digunakan untuk membuka file pada perangkat Core AWS IoT Greengrass ini.
  - Lainnya:
    - CONFIG_POSIX_MQUEUE
    - CONFIG_OVERLAY_FS
    - CONFIG_HAVE_ARCH_SECCOMP_FILTER
    - CONFIG_SECCOMP_FILTER
    - CONFIG_KEYS
    - CONFIG_SECCOMP
    - CONFIG_SHMEM
- Sertifikat root untuk Amazon S3 dan AWS IoT harus ada di penyimpanan kepercayaan sistem.

- Item berikut diperlukan secara kondisional:
  - Perpustakaan yang mendukung AWS Lambda waktu aktif diperlukan oleh fungsi Lambda yang Anda inginkan untuk berjalan secara lokal. Perpustakaan yang diperlukan harus diinstal pada core dan ditambahkan ke variabel lingkungan `PATH` ini. Beberapa perpustakaan dapat diinstal pada Core yang sama.
    - Python versi 2.7 untuk fungsi yang menggunakan waktu aktif Python 2.7.
    - Node.js versi 6.10 atau yang lebih baru untuk fungsi yang menggunakan waktu aktif Node.js 6.10.
    - Java versi 8 atau yang lebih baru untuk fungsi yang menggunakan Java 8 waktu aktif.

- Perintah shell berikut (bukan BusyBox variannya) diperlukan oleh agen pembaruan over-the-air (OTA):
  - `wget`
  - `realpath`
  - `tar`
  - `readlink`
  - `basename`
  - `dirname`
  - `pidof`
  - `df`
  - `grep`
  - `umount`
  - `mv`
  - `gzip`
  - `mkdir`
  - `rm`
  - `ln`
  - `cut`
  - `cat`

Untuk informasi tentang AWS IoT Greengrass kuota (batas), lihat Service Quotas di. Referensi Umum Amazon Web Services

Untuk informasi harga, lihat AWS IoT Greengrass Harga dan AWS IoT Core Harga.

AWS IoT Greengrass unduh

Anda dapat menggunakan informasi berikut untuk menemukan dan mengunduh perangkat lunak untuk digunakan dengan AWS IoT Greengrass.

Topik

- AWS IoT Greengrass Perangkat lunak Core
- AWS IoT Greengrass perangkat lunak snap

- AWS IoT Greengrass Perangkat lunak Docker
- AWS IoT Greengrass Core SDK
- Waktu aktif dan perpustakaan machine learning yang didukung
- AWS IoT Greengrass Perangkat lunak ML SDK

AWS IoT Greengrass Perangkat lunak Core

Perangkat lunak Core AWS IoT Greengrass memperluas fungsionalitas AWS ke perangkat core AWS IoT Greengrass ini, sehingga memungkinkan perangkat lokal bertindak secara lokal pada data yang mereka hasilkan.

v1.11

1.11.6

Perbaikan bug dan peningkatan:

- Peningkatan ketahanan jika kehilangan daya mendadak terjadi selama penyebaran.
- Memperbaiki masalah di mana kerusakan data manajer aliran dapat mencegah perangkat lunak AWS IoT Greengrass Core dimulai.
- Memperbaiki masalah di mana perangkat klien baru tidak dapat terhubung ke inti dalam skenario tertentu.
- Memperbaiki masalah di mana nama aliran pengelola aliran tidak dapat berisi `.log`.

1.11.5

Perbaikan bug dan peningkatan:

- Peningkatan performa umum dan perbaikan bug.

1.11.4

Perbaikan bug dan peningkatan:

- Memperbaiki masalah dengan pengelola aliran yang mencegah peningkatan ke perangkat lunak AWS IoT Greengrass Core v1.11.3. Jika Anda menggunakan stream manager untuk mengekspor data ke cloud, Anda sekarang dapat menggunakan pembaruan OTA untuk memutakhirkan versi v1.x sebelumnya dari perangkat lunak AWS IoT Greengrass Core ke v1.11.4.
- Peningkatan performa umum dan perbaikan bug.

1.11.3

Perbaikan bug dan peningkatan:

- Memperbaiki masalah yang menyebabkan perangkat lunak Core AWS IoT Greengrass berjalan dalam snap pada perangkat Ubuntu yang kemudian berhenti merespons setelah tiba-tiba kehilangan daya pada perangkat.
- Memperbaiki masalah yang menyebabkan keterlambatan pengiriman pesan MQTT ke fungsi Lambda berumur panjang.
- Memperbaiki masalah yang menyebabkan pesan MQTT tidak dikirim dengan benar saat nilai `maxWorkItemCount` telah diatur ke nilai yang lebih besar dari `1024`.
- Memperbaiki masalah yang menyebabkan agen pembaruan OTA mengabaikan MQTT periode `KeepAlive` yang ditentukan dalam properti `keepAlive` di `config.json`.
- Peningkatan performa umum dan perbaikan bug.

> **Important**
>
> Jika Anda menggunakan pengelola aliran untuk mengekspor data ke cloud, jangan tingkatkan ke perangkat lunak AWS IoT Greengrass Core v1.11.3 dari versi v1.x sebelumnya. Jika Anda mengaktifkan pengelola aliran untuk pertama kalinya, kami sangat menyarankan Anda menginstal versi terbaru perangkat lunak AWS IoT Greengrass Core terlebih dahulu.

1.11.1

Perbaikan bug dan peningkatan:

- Memperbaiki masalah yang menyebabkan peningkatan penggunaan memori untuk pengelola aliran.
- Memperbaiki masalah yang menyebabkan pengelola aliran mengatur ulang nomor urut aliran `0` jika perangkat inti Greengrass dimatikan lebih lama dari periode data aliran yang time-to-live ditentukan (TTL).
- Memperbaiki masalah yang mencegah pengelola aliran menghentikan upaya coba lagi dengan benar untuk mengekspor data ke AWS Cloud.

1.11.0

Fitur-fitur baru:

- Agen telemetri pada core Greengrass mengumpulkan data telemetri lokal dan menerbitkannya ke AWS Cloud. Untuk mengambil data telemetri untuk diproses lebih lanjut, pelanggan dapat membuat EventBridge aturan Amazon dan berlangganan target. Untuk informasi lebih lanjut, lihat Mengumpulkan data telemetri kondisi sistem dari perangkat AWS IoT Greengrass core.
- API HTTP lokal mengembalikan snapshot dari keadaan saat ini dari proses pekerja lokal dimulai dengan AWS IoT Greengrass. Untuk informasi lebih lanjut, lihat Memanggil API pemeriksaan kondisi lokal.
- Sebuah Pengelola aliran secara otomatis mengekspor data ke Amazon S3 dan AWS IoT SiteWise.

  Parameter pengelola aliran baru memungkinkan Anda memperbarui aliran yang ada dan menjeda atau melanjutkan ekspor data.
- Support untuk menjalankan fungsi Python 3.8.x Lambda pada core.
- Properti `ggDaemonPort` baru di `config.json` yang digunakan untuk mengonfigurasi nomor port IPC core Greengrass. Nomor port default adalah 8000.

  Properti `systemComponentAuthTimeout` baru di `config.json` yang Anda gunakan untuk mengonfigurasi timeout untuk autentikasi IPC core Greengrass. Timeout default adalah 5000 milidetik.
- Meningkatkan jumlah maksimum perangkat AWS IoT setiap grup AWS IoT Greengrass dari 200 ke 2500.

  Meningkatkan jumlah langganan maksimum per grup dari 1000 menjadi 10000.

  Untuk informasi lebih lanjut, lihat AWS IoT Greengrass kuota dan titik akhir.

Perbaikan bug dan peningkatan:

- Optimasi umum yang dapat mengurangi pemanfaatan memori dari proses layanan Greengrass.
- Parameter konfigurasi waktu aktif baru (`mountAllBlockDevices`) memungkinkan Greengrass menggunakan ikatan pasang untuk memasang semua perangkat blok ke dalam kontainer setelah mengatur OverlayFS. Fitur ini memecahkan masalah yang menyebabkan kegagalan deployment Greengrass jika `/usr` tidak berada di bawah hierarki `/` ini.
- Memperbaiki masalah yang menyebabkan kegagalan core AWS IoT Greengrass jika `/tmp` adalah symlink.

- Memperbaiki masalah untuk membiarkan agen deployment Greengrass menghapus artefak model machine learning yang tidak digunakan dari folder `mlmodel_public` ini.
- Peningkatan performa umum dan perbaikan bug.

Untuk menginstal perangkat lunak AWS IoT Greengrass Core pada perangkat inti Anda, unduh paket untuk arsitektur dan sistem operasi (OS) Anda, lalu ikuti langkah-langkah dalam Panduan Memulai.

> **Tip**
>
> AWS IoT Greengrass juga menyediakan opsi lain untuk menginstal Perangkat lunak Core AWS IoT Greengrass ini. Sebagai contoh, Anda dapat menggunakan Penyiapan perangkat Greengrass untuk mengonfigurasi lingkungan Anda dan menginstal versi terbaru perangkat lunak Core AWS IoT Greengrass ini. Atau, pada platform Debian yang mendukung, Anda dapat menggunakan Manajer paket APT untuk menginstal atau meningkatkan perangkat lunak Core AWS IoT Greengrass ini. Untuk informasi selengkapnya, lihat the section called “Instal AWS IoT Greengrass perangkat lunak Core”.

Arsitektur	Sistem operasi	Tautan
Armv8 (AArch64)	Linux	Unduh
Armv8 (AArch64)	Linux (OpenWrt)	Unduh
Armv7l	Linux	Unduh
Armv7l	Linux (OpenWrt)	Unduh
Armv6l	Linux	Unduh
x86_64	Linux	Unduh

Extended life versions

1.10.5

Fitur baru di v1.10:

- Pengelola aliran yang memproses aliran data secara lokal dan mengekspornya ke AWS Cloud secara otomatis. Fitur ini memerlukan Java 8 pada perangkat Core Greengrass. Untuk informasi selengkapnya, lihat Mengelola aliran data.
- Sebuah konektor deployment aplikasi Greengrass Docker baru yang menjalankan aplikasi Docker pada perangkat Core. Untuk informasi selengkapnya, lihat the section called "Deployment aplikasi Docker".
- SiteWise Konektor IoT baru yang mengirimkan data perangkat industri dari server OPC-UA ke properti aset di. AWS IoT SiteWise Untuk informasi selengkapnya, lihat the section called "IoT SiteWise".
- fungsi Lambda yang berjalan tanpa kontainerisasi dapat mengakses sumber daya machine learning dalam grup Greengrass. Untuk informasi selengkapnya, lihat the section called "Mengakses sumber daya machine learning".
- Support untuk sesi tetap MQTT dengan AWS IoT. Untuk informasi selengkapnya, lihat the section called "Sesi persisten MQTT dengan AWS IoT Core".
- Lalu lintas MQTT lokal dapat melakukan perjalanan melalui port selain port default 8883. Untuk informasi selengkapnya, lihat the section called "Port MQTT untuk olahpesan lokal".
- Pilihan `queueFullPolicy` baru di AWS IoT Greengrass Core SDK untuk penerbitan pesan terpercaya dari fungsi Lambda.
- Support untuk menjalankan Node.js 12.x Lambda fungsi pada core.

Perbaikan bug dan peningkatan:

- Pembaruan O ver-the-air (OTA) dengan integrasi keamanan perangkat keras dapat dikonfigurasi dengan OpenSSL 1.1.
- Pengelola aliran lebih tahan terhadap korupsi data file.
- Memperbaiki masalah yang menyebabkan kegagalan memasang sysfs pada perangkat yang menggunakan kernel Linux 5.1 dan yang lebih baru.
- Properti `mqttOperationTimeout` baru di config.json yang Anda gunakan untuk mengatur timeout agar mempublikasi, berlangganan, dan berhenti berlangganan operasi di koneksi MQTT dengan AWS IoT Core.
- Memperbaiki masalah yang menyebabkan peningkatan penggunaan memori untuk pengelola aliran.
- Properti `systemComponentAuthTimeout` baru di `config.json` yang Anda gunakan untuk mengonfigurasi timeout untuk autentikasi IPC core Greengrass. Timeout default adalah 5000 milidetik.

- Memperbaiki masalah yang menyebabkan agen pembaruan OTA mengabaikan MQTT periode `KeepAlive` yang ditentukan dalam properti `keepAlive` di `config.json`.
- Memperbaiki masalah yang menyebabkan pesan MQTT tidak dikirim dengan benar saat nilai `maxWorkItemCount` telah diatur ke nilai yang lebih besar dari `1024`.
- Memperbaiki masalah yang menyebabkan keterlambatan pengiriman pesan MQTT ke fungsi Lambda berumur panjang.
- Memperbaiki masalah yang menyebabkan perangkat lunak Core AWS IoT Greengrass berjalan dalam snap pada perangkat Ubuntu yang kemudian berhenti merespons setelah tiba-tiba kehilangan daya pada perangkat.
- Peningkatan performa umum dan perbaikan bug.

Untuk menginstal perangkat lunak AWS IoT Greengrass Core pada perangkat inti Anda, unduh paket untuk arsitektur dan sistem operasi (OS) Anda, lalu ikuti langkah-langkah dalam Panduan Memulai.

Arsitektur	Sistem operasi	Tautan
Armv8 (AArch64)	Linux	Unduh
Armv8 (AArch64)	Linux (OpenWrt)	Unduh
Armv7l	Linux	Unduh
Armv7l	Linux (OpenWrt)	Unduh
Armv6l	Linux	Unduh
x86_64	Linux	Unduh

1.9.4

Fitur baru di v1.9:

- Support untuk Python 3.7 dan Node.js 8.10 Lambda waktu aktif. Fungsi Lambda yang menggunakan Python 3.7 dan Node.js 8.10 waktu aktif sekarang dapat berjalan pada core AWS IoT Greengrass ini. (AWS IoT Greengrass terus mendukung Python 2.7 dan Node.js 6.10 waktu aktif.)

- Koneksi MQTT yang dioptimalkan. Core Greengrass menetapkan koneksi yang lebih sedikit dengan AWS IoT Core. Perubahan ini dapat mengurangi biaya operasional untuk beban yang didasarkan pada jumlah koneksi.
- Kunci Elliptic Curve (EC) untuk server MQTT lokal. Server MQTT lokal mendukung kunci EC selain kunci RSA. (Sertifikat server MQTT memiliki tanda tangan SHA-256 RSA, terlepas dari jenis kunci.) Untuk informasi selengkapnya, lihat the section called “Keamanan utama”.
- Support untuk OpenWrt. AWS IoT Greengrass Perangkat lunak inti v1.9.2 atau yang lebih baru dapat diinstal pada OpenWrt distribusi dengan arsitektur Armv8 (AArch64) dan ARMv7L. Saat ini, OpenWrt tidak mendukung inferensi ML.
- Support untuk Armv6l. AWS IoT Greengrass Perangkat lunak Core v1.9.3 atau yang lebih baru dapat diinstal pada distribusi Raspbian dengan arsitektur Armv6l (sebagai contoh, pada perangkat Raspberry Pi Zero).
- OTA membarui pada port 443 dengan ALPN. Greengrass core yang menggunakan port 443 untuk lalu lintas MQTT sekarang mendukung pembaruan perangkat lunak (OTA). over-the-air AWS IoT Greengrassmenggunakan ekstensi Application Layer Protocol Network (ALPN) TLS untuk mengaktifkan koneksi ini. Lihat informasi yang lebih lengkap di Perbarui OTA AWS IoT Greengrass Perangkat lunak Core dan the section called “Connect pada port 443 atau melalui proksi jaringan”.

Untuk menginstal perangkat lunak AWS IoT Greengrass Core pada perangkat inti Anda, unduh paket untuk arsitektur dan sistem operasi (OS) Anda, lalu ikuti langkah-langkah dalam Panduan Memulai.

Arsitektur	Sistem operasi	Tautan
Armv8 (AArch64)	Linux	Unduh
Armv8 (AArch64)	Linux (OpenWrt)	Unduh
Armv7l	Linux	Unduh
Armv7l	Linux (OpenWrt)	Unduh
Armv6l	Linux	Unduh
x86_64	Linux	Unduh

1.8.4

- Fitur-fitur baru:
  - Identitas akses default yang dapat dikonfigurasi untuk fungsi Lambda dalam grup. Pengaturan tingkat grup ini menentukan izin default yang digunakan untuk menjalankan fungsi Lambda. Anda dapat mengatur ID pengguna, ID grup, atau keduanya. Fungsi Lambda individu dapat menimpa identitas akses default grup mereka. Untuk informasi selengkapnya, lihat the section called “Mengatur identitas akses default untuk fungsi Lambda dalam grup”.
  - Lalu lintas HTTPS melalui port 443. Komunikasi HTTPS dapat dikonfigurasi untuk perjalanan melalui port 443 daripada port default 8443. Ini melengkapi dukungan AWS IoT Greengrass untuk ekstensi TLS Application Layer Protocol Network (ALPN) dan mengizinkan semua lalu lintas olahpesan Greengrass—baik MQTT maupun HTTPS—untuk menggunakan port 443. Untuk informasi selengkapnya, lihat the section called “Connect pada port 443 atau melalui proksi jaringan”.
  - ID klien yang dapat diprediksi namanya untuk koneksi AWS IoT ini. Perubahan ini mengaktifkan dukungan untuk AWS IoT Device Defender dan AWS IoT Siklus hidup, sehingga Anda dapat menerima notifikasi untuk koneksi, putuskan sambungan, berlangganan, dan berhenti berlangganan acara. Penamaan yang dapat diprediksi juga membuatnya lebih mudah untuk membuat logika di sekitar ID koneksi (sebagai contoh, untuk membuat templat kebijakan langganan berdasarkan atribut sertifikat). Untuk informasi selengkapnya, lihat the section called “ID klien untuk koneksi MQTT dengan AWS IoT”.

  Perbaikan bug dan peningkatan:

  - Memperbaiki masalah dengan sinkronisasi bayangan dan rekoneksi Certificate Manager perangkat.
  - Peningkatan performa umum dan perbaikan bug.

Untuk menginstal perangkat lunak AWS IoT Greengrass Core pada perangkat inti Anda, unduh paket untuk arsitektur dan sistem operasi (OS) Anda, lalu ikuti langkah-langkah dalam Panduan Memulai.

Arsitektur	Sistem operasi	Tautan
Armv8 (AArch64)	Linux	Unduh

Arsitektur	Sistem operasi	Tautan
Armv7l	Linux	Unduh
x86_64	Linux	Unduh

Dengan mengunduh perangkat lunak ini, Anda menyetujui Perjanjian Lisensi Perangkat Lunak Core Greengrass.

Untuk informasi tentang opsi lain untuk menginstal perangkat lunak core AWS IoT Greengrass di perangkat Anda, lihat the section called “Instal AWS IoT Greengrass perangkat lunak Core”.

AWS IoT Greengrass perangkat lunak snap

AWS IoT Greengrass snap 1.11.x mengaktifkan Anda untuk menjalankan versi terbatas dari AWS IoT Greengrass melalui paket perangkat lunak yang mudah digunakan, bersama dengan semua dependensi yang diperlukan, dalam lingkungan kontainer.

> **Note**
>
> Snap AWS IoT Greengrass tersedia untuk perangkat lunak Core AWS IoT Greengrass v1.11.x. AWS IoT Greengrass tidak menyediakan snap untuk v1.10.x. Versi yang tidak didukung tidak dapat menerima perbaikan bug atau pembaruan.
> Snap AWS IoT Greengrass tidak mendukung konektor dan inferensi machine learning (ML).

Untuk informasi selengkapnya, lihat the section called “Jalankan AWS IoT Greengrass dalam snap”.

AWS IoT Greengrass Perangkat lunak Docker

AWS menyediakan Dockerfile dan citra Docker yang membuatnya lebih mudah bagi Anda untuk menjalankan AWS IoT Greengrass dalam kontainer Docker.

Dockerfile

Dockerfiles berisi kode sumber untuk membangun citra kontainer AWS IoT Greengrass kustom. Citra dapat dimodifikasi agar berjalan pada arsitektur platform yang berbeda atau untuk mengurangi ukuran citra. Untuk instruksi, lihat file README.

Unduh target Anda versi perangkat lunak core AWS IoT Greengrass ini.

v1.11

- Dockerfile untuk v1.11.6 AWS IoT Greengrass.

Extended life versions

v1.10

Dockerfile untuk v1.10.5 AWS IoT Greengrass.

v1.9

Dockerfile untuk AWS IoT Greengrass v1.9.4.

v1.8

Dockerfile untuk AWS IoT Greengrass v1.8.1.

Citra Docker

Citra Docker memiliki perangkat lunak core AWS IoT Greengrass dan dependensi yang diinstal pada Citra dasar Amazon Linux 2 (x86_64) dan Alpine Linux (x86_64, Armv7l, atau AArch64). Anda dapat menggunakan Citra prebuilt untuk mulai bereksperimen dengan AWS IoT Greengrass.

> **Important**
>
> Pada tanggal 30 Juni 2022, AWS IoT Greengrass mengakhiri pemeliharaan untuk perangkat lunak AWS IoT Greengrass Core v1.x gambar Docker yang diterbitkan ke Amazon Elastic Container Registry (Amazon ECR) Registry ECR) dan Docker Hub. Anda dapat terus mengunduh gambar Docker ini dari Amazon ECR dan Docker Hub hingga 30 Juni 2023, yaitu 1 tahun setelah pemeliharaan berakhir. Namun, gambar AWS IoT Greengrass Core software v1.x Docker tidak lagi menerima tambalan keamanan atau perbaikan bug setelah pemeliharaan berakhir pada 30 Juni 2022. Jika Anda menjalankan beban kerja produksi yang bergantung pada gambar Docker ini, kami sarankan Anda

membuat gambar Docker Anda sendiri menggunakan Dockerfiles yang menyediakan. AWS IoT Greengrass Untuk informasi selengkapnya, lihat AWS IoT Greengrass Version 1kebijakan pemeliharaan.

Unduh Citra prebuilt dari Docker Hub atau Amazon Elastic Container Registry (Amazon ECR).

- Untuk Docker Hub, gunakan tag *`versi`* untuk mengunduh versi tertentu dari citra Docker Greengrass. Untuk menemukan tag pada semua Citra yang tersedia, periksa Tag di Docker Hub.
- Untuk Amazon ECR, gunakan tag `latest` untuk mengunduh versi terbaru yang tersedia dari citra Docker Greengrass. Untuk informasi lebih lanjut tentang daftar versi citra yang tersedia dan mengunduh citra dari Amazon ECR, lihat Menjalankan AWS IoT Greengrass di kontainer Docker.

Warning

Dimulai dengan v1.11.6 dari perangkat lunak AWS IoT Greengrass Core, gambar Greengrass Docker tidak lagi menyertakan Python 2.7, karena Python 2.7 mencapai pada tahun 2020 dan tidak lagi menerima pembaruan keamanan. end-of-life Jika Anda memilih untuk memperbarui ke gambar Docker ini, kami sarankan Anda memvalidasi bahwa aplikasi Anda bekerja dengan gambar Docker baru sebelum Anda menyebarkan pembaruan ke perangkat produksi. Jika Anda memerlukan Python 2.7 untuk aplikasi Anda yang menggunakan image Greengrass Docker, Anda dapat memodifikasi Greengrass Dockerfile untuk menyertakan Python 2.7 untuk aplikasi Anda.

AWS IoT Greengrass tidak menyediakan citra Docker untuk perangkat lunak core AWS IoT Greengrass v1.11.1.

Note

Secara default, citra `alpine-aarch64` dan `alpine-armv7l` dapat berjalan hanya pada host berbasis ARM. Untuk menjalankan citra ini pada host x86, Anda dapat menginstal QEMU dan memasang perpustakaan QEMU di host. Sebagai contoh:

```
docker run --rm --privileged multiarch/qemu-user-static --reset -p yes
```

AWS IoT Greengrass Core SDK

Menggunakan fungsi Lambda AWS IoT Greengrass Core SDK untuk berinteraksi dengan core AWS IoT Greengrass ini. Hal ini mengizinkan fungsi Lambda di-deploy untuk:

- Pertukaran pesan MQTT dengan AWS IoT Core.
- Tukarkan pesan MQTT dengan konektor, perangkat klien, dan fungsi Lambda lainnya di grup Greengrass.
- Berinteraksi dengan layanan bayangan lokal.
- Meminta fungsi Lambda lokal lainnya.
- Akses Sumber daya rahasia.
- Berinteraksi dengan Pengelola aliran.

Unduh AWS IoT Greengrass Core SDK untuk bahasa atau platform Anda dari GitHub.

- AWS IoT GreengrassCore SDK for Java
- AWS IoT GreengrassCore SDK untuk Node.js
- AWS IoT GreengrassCore SDK untuk Python
- AWS IoT GreengrassCore SDK untuk C

Untuk informasi selengkapnya, lihat AWS IoT GreengrassSDK inti.

Waktu aktif dan perpustakaan machine learning yang didukung

Untuk lakukan inferensi pada core Greengrass, Anda harus menginstal waktu aktif machine learning atau perpustakaan untuk jenis model ML Anda.

AWS IoT Greengrass mendukung jenis model ML berikut. Gunakan tautan ini untuk menemukan informasi tentang cara menginstal waktu aktif atau perpustakaan untuk jenis model dan platform perangkat Anda.

- Runtime Pembelajaran Mendalam (DLR)

- MxNet
- TensorFlow

Sampel machine learning

AWS IoT Greengrass menyediakan sampel yang dapat Anda gunakan dengan waktu aktif ML yang didukung dan perpustakaan. Sampel ini dirilis menurut Perjanjian Lisensi Perangkat Lunak Core Greengrass.

Deep learning runtime (DLR)

Unduh contoh untuk platform perangkat Anda:

- Sampel DLR untuk Raspberry Pi
- Sampel DLR untuk NVIDIA Jetson TX2
- Sampel DLR untuk Intel Atom

Untuk tutorial yang menggunakan sampel DLR, lihat the section called “Cara mengonfigurasi kesimpulan machine learning yang dioptimalkan”.

MXNet

Unduh contoh untuk platform perangkat Anda:

- Sampel MXNet untuk Raspberry Pi
- Sampel MXNet untuk NVIDIA Jetson TX2
- Sampel MXNet untuk Intel Atom

Untuk tutorial yang menggunakan sampel MXNet, lihat the section called “Cara mengonfigurasi inferensi machine learning”.

TensorFlow

Unduh Sampel Tensorflow untuk platform perangkat Anda. Sampel ini bekerja dengan Raspberry Pi, NVIDIA Jetson TX2, dan Intel Atom.

AWS IoT Greengrass Perangkat lunak ML SDK

Ini AWS IoT GreengrassSDK Machine Learning memungkinkan fungsi Lambda yang Anda tulis untuk mengonsumsi model machine learning lokal dan mengirim data ke konektor ML Feedback untuk mengunggah dan menerbitkan.

v1.1.0

- Python 3.7.

v1.0.0

- Python 2.7.

Kami ingin mendengar pendapat Anda

Kami menyambut umpan balik Anda. Untuk menghubungi kami, kunjungi AWSre:Post dan gunakan tag. AWS IoT Greengrass

Instal AWS IoT Greengrass perangkat lunak Core

Perangkat lunak Core AWS IoT Greengrass memperluas fungsionalitas AWS ke AWS IoT Greengrass perangkat core, sehingga memungkinkan perangkat lokal bertindak secara lokal pada data yang mereka hasilkan.

AWS IoT Greengrass menyediakan beberapa opsi untuk menginstal AWS IoT Greengrass perangkat lunak Core:

- mengunduh dan mengekstraksi file tar.gz.
- Jalankan skrip Pengaturan Perangkat Greengrass.
- Instal dari repositori APT.

AWS IoT Greengrass juga menyediakan lingkungan terkontainerisasi yang menjalankan AWS IoT Greengrass perangkat lunak Core.

- Jalankan AWS IoT Greengrass dalam kontainer Docker.
- Jalankan AWS IoT Greengrass dalam snap.

Mengunduh dan mengekstraksi AWS IoT Greengrass paket perangkat lunak Core

Memilih AWS IoT Greengrass perangkat lunak core untuk platform Anda untuk diunduh sebagai file tar.gz dan mengekstraksi pada perangkat Anda. Anda dapat mengunduh versi terbaru dari perangkat lunak. Untuk informasi selengkapnya, lihat the section called “AWS IoT Greengrass Perangkat lunak Core”.

Jalankan skrip pengaturan perangkat Greengrass

Jalankan pengaturan perangkat Greengrass untuk mengonfigurasi perangkat Anda, menginstal versi AWS IoT Greengrass terbaru perangkat lunak Core, dan men-deploy fungsi Hello World Lambda dalam hitungan menit. Untuk informasi selengkapnya, lihat the section called “Quick start: penyiapan perangkat Greengrass”.

Instal AWS IoT Greengrass perangkat lunak core dari repositori APT

> **Important**
>
> Mulai 11 Februari 2022, Anda tidak dapat lagi menginstal atau memperbarui perangkat lunak AWS IoT Greengrass Core dari repositori APT. Pada perangkat tempat Anda menambahkan AWS IoT Greengrass repositori, Anda harus menghapus repositori dari daftar sumber. Perangkat yang menjalankan perangkat lunak dari repositori APT akan terus beroperasi secara normal. Kami menyarankan Anda memperbarui perangkat lunak AWS IoT Greengrass Core menggunakan file tar.

Repositori APT yang disediakan oleh AWS IoT Greengrass termasuk paket berikut:

- `aws-iot-greengrass-core`. Menginstal AWS IoT Greengrass perangkat lunak Core.
- `aws-iot-greengrass-keyring`. Menginstal tombol GnuPG (GPG) yang digunakan untuk menandatangani AWS IoT Greengrass paket repositori.

Dengan mengunduh perangkat lunak ini, Anda menyetujui Perjanjian Lisensi Perangkat Lunak Greengrass Core.

Topik

- Gunakan skrip systemd untuk mengelola siklus hidup Greengrass daemon
- Copot pemasangan perangkat lunak AWS IoT Greengrass inti menggunakan repositori APT
- Hapus sumber repositori perangkat lunak AWS IoT Greengrass inti

Gunakan skrip systemd untuk mengelola siklus hidup Greengrass daemon

Paket `aws-iot-greengrass-core` juga menginstal `systemd` skrip yang dapat Anda gunakan untuk mengelola AWS IoT Greengrass siklus hidup perangkat lunak Core (daemon).

- Untuk memulai Greengrass daemon saat boot:

```
systemctl enable greengrass.service
```

- Untuk memulai Greengrass daemon:

```
systemctl start greengrass.service
```

- Untuk menghentikan Greengrass daemon:

```
systemctl stop greengrass.service
```

- Untuk memeriksa status Greengrass daemon:

```
systemctl status greengrass.service
```

Copot pemasangan perangkat lunak AWS IoT Greengrass inti menggunakan repositori APT

Ketika Anda menghapus perangkat lunak AWS IoT Greengrass inti, Anda dapat memilih apakah akan menyimpan atau menghapus informasi konfigurasi perangkat lunak AWS IoT Greengrass inti, seperti sertifikat perangkat, informasi grup, dan file log.

Untuk menghapus perangkat lunak AWS IoT Greengrass inti dan melestarikan informasi konfigurasi

- Jalankan perintah berikut untuk menghapus paket perangkat lunak AWS IoT Greengrass inti dan menyimpan informasi konfigurasi dalam `/greengrass` folder.

```
sudo apt remove aws-iot-greengrass-core aws-iot-greengrass-keyring
```

Untuk menghapus perangkat lunak AWS IoT Greengrass inti dan menghapus informasi konfigurasi

1. Jalankan perintah berikut untuk menghapus paket perangkat lunak AWS IoT Greengrass inti dan menghapus informasi konfigurasi dari file`/greengrass folder`.

```
sudo apt purge aws-iot-greengrass-core aws-iot-greengrass-keyring
```

2. Hapus repositori perangkat lunak AWS IoT Greengrass inti dari daftar sumber Anda. Untuk informasi selengkapnya, lihat Hapus sumber repositori perangkat lunak AWS IoT Greengrass inti.

Hapus sumber repositori perangkat lunak AWS IoT Greengrass inti

Anda dapat menghapus sumber repositori perangkat lunak AWS IoT Greengrass inti ketika Anda tidak perlu lagi menginstal atau memperbarui perangkat lunak AWS IoT Greengrass inti dari repositori APT. Setelah 11 Februari 2022, Anda harus menghapus repositori dari daftar sumber Anda untuk menghindari kesalahan saat menjalankan. `apt update`

Untuk menghapus repositori APT dari daftar sumber

- Jalankan perintah berikut untuk menghapus repositori perangkat lunak AWS IoT Greengrass inti dari daftar sumber.

```
sudo rm /etc/apt/sources.list.d/greengrass.list
sudo apt update
```

Jalankan AWS IoT Greengrass dalam kontainer Docker

AWS IoT Greengrass menyediakan Dockerfile dan gambar Docker yang membuatnya lebih mudah bagi Anda untuk menjalankan AWS IoT Greengrass perangkat lunak Core dalam kontainer Docker.

Untuk informasi selengkapnya, lihat the section called “AWS IoT Greengrass Perangkat lunak Docker”.

> **Note**
>
> Anda juga dapat menjalankan aplikasi Docker pada perangkat core Greengrass. Untuk melakukannya, gunakan Greengrass Docker aplikasi deployment konektor.

Jalankan AWS IoT Greengrass dalam snap

AWS IoT Greengrass snap 1.11.x memungkinkan Anda untuk menjalankan versi terbatas AWS IoT Greengrass melalui paket perangkat lunak yang mudah digunakan, bersama dengan semua dependensi yang diperlukan, dalam lingkungan terkontainer.

Pada 31 Desember 2023, AWS IoT Greengrass akan mengakhiri pemeliharaan untuk perangkat lunak AWS IoT Greengrass inti versi 1.11.x Snap yang diterbitkan di snapcraft.io. Perangkat yang saat ini menjalankan Snap akan terus berfungsi hingga pemberitahuan lebih lanjut. Namun, Snap AWS IoT Greengrass inti tidak akan lagi menerima tambalan keamanan atau perbaikan bug setelah pemeliharaan berakhir.

Konsep snap

Berikut ini adalah konsep snap penting untuk membantu Anda memahami bagaimana menggunakan AWS IoT Greengrass snap:

Kanal

Komponen snap yang mendefinisikan versi snap yang diinstal dan dilacak untuk update. Snap di-update secara otomatis ke versi terbaru saluran saat ini.

Antarmuka

Komponen snap yang memberikan akses ke sumber daya, seperti jaringan dan file pengguna.

Untuk menjalankan AWS IoT Greengrass snap, antaramuka berikut harus terhubung. Perhatikan bahwa `greengrass-support-no-container` harus terhubung terlebih dahulu dan tidak pernah terputus.

```
    - greengrass-support-no-container
    - hardware-observe
    - home-for-hooks
    - hugepages-control
    - log-observe
    - mount-observe
    - network
    - network-bind
    - network-control
    - process-control
    - system-observe
```

Antarmuka lain bersifat opsional. Jika fungsi Lambda Anda memerlukan akses ke sumber daya tertentu, Anda mungkin perlu terhubung ke antarmuka yang sesuai.

Segarkan

Snap di-update secara otomatis. Daemon `snapd` adalah pengelola paket snap yang memeriksa pembaruan empat kali sehari secara default. Setiap update memeriksa disebut refresh. Ketika refresh terjadi, daemon berhenti, snap akan di-update, dan kemudian daemon restart.

Untuk informasi lebih lanjut, lihat situs web Snapcraft ini.

Yang baru dengan AWS IoT Greengrass snap v1.11.x

Berikut ini menjelaskan apa yang baru dan berubah dengan versi 1.11.x AWS IoT Greengrass snap.

- Versi ini hanya mendukung `snap_daemon` pengguna, diekspos sebagai user ID (UID) dan group (GID) 584788.
- Versi ini hanya mendukung fungsi Lambda noncontainerized.

> **Important**
>
> Karena fungsi Lambda noncontainerized harus berbagi pengguna yang sama (`snap_daemon`), fungsi Lambda tidak memiliki isolasi satu sama lain. Untuk informasi selengkapnya, lihat Mengontrol pelaksanaan fungsi Greengrass Lambda dengan menggunakan konfigurasi khusus grup.

- Versi ini mendukung C, C ++, Java 8, Node.js 12.x, Python 2.7, Python 3.7, dan Python 3.8 waktu aktif.

> **Note**
>
> Untuk menghindari waktu aktif Python berlebihan, Python 3.7 fungsi Lambda benar-benar menjalankan Python 3.8 waktu aktif.

Memulai dengan AWS IoT Greengrass snap

Prosedur berikut membantu Anda menginstal dan mengonfigurasi AWS IoT Greengrass snap pada perangkat anda.

Persyaratan

Untuk menjalankan AWS IoT Greengrass snap, Anda harus melakukan hal berikut:

- Jalankan AWS IoT Greengrass snap pada distribusi Linux yang didukung, seperti Ubuntu, Linux Mint, Debian, dan Fedora.
- Instal `snapd` daemon pada perangkat Anda. Daemon `snapd` termasuk `snap` alat mengelola lingkungan snap pada perangkat Anda.

Untuk daftar distribusi Linux yang didukung dan petunjuk penginstalan, lihat Menginstal snapd dalam Dokumentasi Snap.

Instal dan konfigurasi AWS IoT Greengrass snap

Tutorial berikut menunjukkan kepada Anda bagaimana menginstal dan mengonfigurasi AWS IoT Greengrass snap pada perangkat Anda.

> **Note**
>
> - Meskipun tutorial ini menggunakan contoh Amazon EC2 (x86 t2.micro Ubuntu 20.04), Anda dapat menjalankan AWS IoT Greengrass snap dengan prangkat keras fisik, seperti Raspberry Pi.
> - Daemon `snapd` sudah terinstal di Ubuntu.

1. Instal `core18` snap dengan menjalankan perintah berikut di terminal perangkat Anda:

```
sudo snap install core18
```

Snap `core18` adalah snap dasar yang menyediakan lingkungan waktu aktif dengan perpustakaan umum yang digunakan. Snap ini dibangun dari LTS Ubuntu 18.04.

2. Upgrade `snapd` dengan menjalankan perintah berikut:

```
sudo snap install --channel=edge snapd; sudo snap refresh --channel=edge snapd
```

3. Jalankan `snap list` perintah untuk memeriksa apakah Anda telah AWS IoT Greengrass menginstal snap.

   Contoh respons berikut menunjukkan bahwa `snapd` dipasang, tetapi `aws-iot-greengrass` bukan.

```
Name              Version                Rev    Tracking         Publisher   Notes
amazon-ssm-agent  3.0.161.0              2996   latest/stable/…  aws#        classic
core              16-2.48                10444  latest/stable    canonical#  core
core18            20200929               1932   latest/stable    canonical#  base
lxd               4.0.4                  18150  4.0/stable/…     canonical#  -
snapd             2.48+git548.g929ccfb   10526  latest/edge      canonical#  snapd
```

4. Pilih salah satu opsi berikut untuk diinstal AWS IoT Greengrass snap 1.11.x.

   - Untuk instal AWS IoT Greengrass di Ubuntu, jalankan perintah berikut:

```
sudo snap install aws-iot-greengrass
```

     Contoh respons:

```
aws-iot-greengrass 1.11.5 from Amazon Web Services (aws) installed
```

   - Untuk bermigrasi dari versi sebelumnya ke v1.11.x atau update ke versi patch terbaru yang tersedia, jalankan perintah berikut:

```
sudo snap refresh --channel=1.11.x aws-iot-greengrass
```

Seperti snap lainnya, AWS IoT Greengrass snap menggunakan saluran untuk mengelola versi minor. Snap di-update secara otomatis ke versi terbaru saluran saat ini yang tersedia. Misalnya,

jika Anda menentukan`--channel=1.11.x`, AWS IoT Greengrass snap Anda diperbarui ke v1.11.5.

Anda dapat menjalankan `snap info aws-iot-greengrass` untuk mendapatkan daftar saluran yang tersedia untuk AWS IoT Greengrass.

Contoh respons:

```
name:      aws-iot-greengrass
summary:   AWS supported software that extends cloud capabilities to local devices.
publisher: Amazon Web Services (aws#)
store-url: https://snapcraft.io/aws-iot-greengrass
contact:   https://repost.aws/tags/TA4ckIed1sR4enZBey29rKTg/aws-io-t-greengrass
license:   Proprietary
description: |
  AWS IoT Greengrass seamlessly extends AWS onto edge devices so they can act
 locally on the data
  they generate, while still using the cloud for management, analytics, and durable
 storage.
  AWS IoT Greenrgrass snap v1.11.0 enables you to run a limited version of AWS IoT
 Greengrass with
  all necessary dependencies in a containerized environment.
  The AWS IoT Greengrass snap doesn't support connectors and machine learning (ML)
 inference.
  By downloading this software you agree to the Greengrass Core Software License
 Agreement
  (https://s3-us-west-2.amazonaws.com/greengrass-release-license/greengrass-
license-v1.pdf).
  For more information, see Run AWS IoT Greengrass in a snap
  (https://docs.aws.amazon.com/greengrass/latest/developerguide/install-
ggc.html#gg-snap-support) in
  the AWS IoT Greengrass Developer.
  If you need help, try the AWS IoT Greengrass tag on AWS re:Post
  (https://repost.aws/tags/TA4ckIed1sR4enZBey29rKTg/aws-io-t-greengrass) or connect
 with an AWS IQ expert
  (https://iq.aws.amazon.com/services/aws/greengrass).
snap-id: SRDuhPJGj4XPxFNNZQKOTvURAp0wxKnd
channels:
  latest/stable:    1.11.3 2021-06-15 (59) 111MB -
  latest/candidate: 1.11.3 2021-06-14 (59) 111MB -
  latest/beta:      1.11.3 2021-06-14 (59) 111MB -
  latest/edge:      1.11.3 2021-06-14 (59) 111MB -
  1.11.x/stable:    1.11.3 2021-06-15 (59) 111MB -
```

```
  1.11.x/candidate: 1.11.3 2021-06-15 (59) 111MB -
  1.11.x/beta:      1.11.3 2021-06-15 (59) 111MB -
  1.11.x/edge:      1.11.3 2021-06-15 (59) 111MB -
```

5. Untuk mengakses sumber daya spesifik yang dibutuhkan fungsi Lambda Anda, Anda dapat terhubung ke antarmuka tambahan.

   Jalankan perintah berikut untuk mendapatkan daftar AWS IoT Greengrass antarmuka yang didukung snap:

   ```
   snap connections aws-iot-greengrass
   ```

   Contoh respons:

   ```
   Interface               Plug                                            Slot
               Notes
   camera                  aws-iot-greengrass:camera                       -
               -
   dvb                     aws-iot-greengrass:dvb                          -
               -
   gpio                    aws-iot-greengrass:gpio                         -
               -
   gpio-memory-control     aws-iot-greengrass:gpio-memory-control          -
               -
   greengrass-support      aws-iot-greengrass:greengrass-support-no-container
     :greengrass-support  -
   hardware-observe        aws-iot-greengrass:hardware-observe
     :hardware-observe    manual
   hardware-random-control  aws-iot-greengrass:hardware-random-control     -
               -
   home                    aws-iot-greengrass:home-for-greengrassd         -
               -
   home                    aws-iot-greengrass:home-for-hooks               :home
               manual
   hugepages-control       aws-iot-greengrass:hugepages-control
     :hugepages-control   manual
   i2c                     aws-iot-greengrass:i2c                          -
               -
   iio                     aws-iot-greengrass:iio                          -
               -
   joystick                aws-iot-greengrass:joystick                     -
               -
   ```

```
log-observe              aws-iot-greengrass:log-observe                    :log-
observe         manual
mount-observe            aws-iot-greengrass:mount-observe
  :mount-observe       manual
network                  aws-iot-greengrass:network
  :network             -
network-bind             aws-iot-greengrass:network-bind
  :network-bind        -
network-control          aws-iot-greengrass:network-control
  :network-control     -
opengl                   aws-iot-greengrass:opengl
  :opengl              -
optical-drive            aws-iot-greengrass:optical-drive
  :optical-drive       -
process-control          aws-iot-greengrass:process-control
  :process-control     -
raw-usb                  aws-iot-greengrass:raw-usb                        -
               -
removable-media          aws-iot-greengrass:removable-media                -
               -
serial-port              aws-iot-greengrass:serial-port                    -
               -
spi                      aws-iot-greengrass:spi                            -
               -
system-observe           aws-iot-greengrass:system-observe
  :system-observe      -
```

Jika Anda melihat tanda hubung (-) di kolom Slot, antarmuka yang sesuai tidak terhubung.

6. Ikuti Menginstal perangkat lunak AWS IoT Greengrass Core untuk membuat AWS IoT sesuatu, grup Greengrass, sumber daya keamanan yang memungkinkan komunikasi aman AWS IoT dengan, dan AWS IoT Greengrass file konfigurasi perangkat lunak Core. File konfigurasi,`config.json`, berisi konfigurasi khusus untuk inti Greengrass Anda, seperti lokasi file sertifikat dan titik akhir data perangkat. AWS IoT

   > **Note**
   >
   > Jika Anda mengunduh file ke perangkat lain, ikuti langkah ini untuk mentransfer file ke perangkat AWS IoT Greengrass inti.

7. Untuk AWS IoT Greengrass snap, pastikan bahwa Anda meng-update file config.json, seperti yang ditunjukkan dalam berikut:

- Ganti setiap instance *CertificateID* dengan ID sertifikat atas nama sertifikat dan file kunci.
- Jika Anda mengunduh sertifikat CA root Amazon yang berbeda dari Amazon Root CA 1, ganti setiap instance *AmazonRootCa1.PEM* dengan nama file CA root Amazon.

```
{
  ...
  "crypto" : {
    "principals" : {
      "SecretsManager" : {
        "privateKeyPath" : "file:///snap/aws-iot-greengrass/current/greengrass/
certs/certificateId-private.pem.keyy"
      },
      "IoTCertificate" : {
        "privateKeyPath" : "file:///snap/aws-iot-greengrass/current/greengrass/
certs/certificateId-private.pem.key",
        "certificatePath" : "file:///snap/aws-iot-greengrass/current/greengrass/
certs/certificateId-certificate.pem.crt"
      }
    },
    "caPath" : "file:///snap/aws-iot-greengrass/current/greengrass/
certs/AmazonRootCA1.pem"
  },
  "writeDirectory": "/var/snap/aws-iot-greengrass/current/ggc-write-directory",
  "pidFileDirectory": "/var/snap/aws-iot-greengrass/current/pidFileDirectory"
}
```

8. Jalankan perintah berikut untuk menambahkan AWS IoT Greengrass sertifikat dan file konfigurasi:

```
sudo snap set aws-iot-greengrass gg-certs=/home/ubuntu/my-certs
```

Deploying fungsi Lambda

Bagian ini menunjukkan bagaimana untuk men-deploy fungsi Lambda yang dikelola pelanggan pada AWS IoT Greengrass snap.

> **Important**
>
> AWS IoT Greengrass snap v1.11 hanya mendukung fungsi Lambda noncontainerized.

1. Jalankan perintah berikut untuk memulai AWS IoT Greengrass daemon:

   ```
   sudo snap start aws-iot-greengrass
   ```

   Contoh respons:

   ```
   Started.
   ```

   > **Note**
   >
   > Jika Anda mendapatkan kesalahan, Anda dapat menggunakan perintah `snap run` untuk pesan kesalahan terperinci. Untuk informasi pemecahan masalah selengkapnya, lihat error: tidak dapat melakukan tugas-tugas berikut: - Jalankan perintah layanan "start" untuk layanan ["greengrassd"] dari snap "aws-iot-greengrass" ([start snap. aws-iot-greengrass.greengrassd.service] gagal dengan status keluar 1: Job for snap. aws-iot-greengrass.greengrassd.service gagal karena proses kontrol keluar dengan kode kesalahan. Lihat "snap status systemctl. aws-iot-greengrass.greengrassd.service" dan "journalctl -xe" untuk detailnya.).

2. Jalankan perintah berikut untuk mengonfirmasi bahwa daemon sedang berjalan:

   ```
   snap services aws-iot-greengrass.greengrassd
   ```

   Contoh respons:

   ```
   Service                          Startup   Current  Notes
   aws-iot-greengrass.greengrassd   disabled  active   -
   ```

3. Ikuti Modul 3 (bagian 1): Lambda berfungsi pada AWS IoT Greengrass untuk membuat dan men-deploy fungsi Hello World Lambda. Namun, sebelum Anda men-deploy fungsi Lambda, selesaikan langkah berikutnya.

4. Pastikan bahwa fungsi Lambda Anda menjalankan sebagai pengguna `snap_daemon` dan dalam mode tanpa kontainer. Untuk update pengaturan grup Greengrass Anda, lakukan hal berikut di AWS IoT Greengrass konsol:

   a. Masuk ke AWS IoT Greengrass konsol.
   b. Di panel navigasi AWS IoT konsol, di bawah Kelola, perluas perangkat Greengrass, lalu pilih Grup (V1).
   c. Di bawah Grup Greengrass, pilih grup target.
   d. Pada halaman konfigurasi grup, di panel navigasi, pilih tab Fungsi Lambda.
   e. Di bawah lingkungan runtime fungsi Lambda default, pilih Edit, dan lakukan hal berikut:

      i. Untuk pengguna dan grup sistem default, pilih ID pengguna lain/ID grup, lalu masukkan **584788** untuk ID pengguna sistem (nomor) dan ID grup sistem (nomor).
      ii. Untuk kontainerisasi fungsi Lambda Default, pilih Tanpa kontainer.
      iii. Pilih Simpan.

Menghentikan AWS IoT Greengrass daemon

Anda dapat menggunakan `snap stop` perintah untuk menghentikan layanan.

Untuk menghentikan AWS IoT Greengrass daemon, jalankan perintah berikut:

```
sudo snap stop aws-iot-greengrass
```

Perintah tersebut seharusnya mengembalikan `Stopped.`.

Untuk memeriksa apakah Anda berhasil menghentikan snap, jalankan perintah berikut:

```
snap services aws-iot-greengrass.greengrassd
```

Contoh respons:

```
Service                         Startup   Current   Notes
aws-iot-greengrass.greengrassd  disabled  inactive  -
```

Menghapus instalasi AWS IoT Greengrass snap

Untuk menghapus instalan AWS IoT Greengrass snap, jalankan perintah berikut:

```
sudo snap remove aws-iot-greengrass
```

Contoh respons:

```
aws-iot-greengrass removed
```

Pemecahan masalah AWS IoT Greengrass snap

Informasi berikut dapat membantu Anda memecahkan masalah dengan AWS IoT Greengrass snap.

Mendapat kesalahan izin ditolak.

Solusi: Kesalahan izin ditolak sering kali karena antarmuka yang hilang. Untuk daftar antarmuka yang hilang dan informasi pemecahan masalah terperinci, Anda dapat menggunakan `snappy-debug` alat.

Jalankan perintah berikut untuk menginstal alat.

```
sudo snap install snappy-debug
```

Contoh respons:

```
snappy-debug 0.36-snapd2.45.1 from Canonical# installed
```

Jalankan `sudo snappy-debug` perintah dalam sesi terminal terpisah. Operasi berlanjut hingga terjadi kesalahan izin ditolak.

Sebagai contoh, jika fungsi Lambda Anda mencoba untuk membaca file di `$HOME` direktori, Anda mungkin mendapatkan respons berikut:

```
INFO: Following '/var/log/syslog'. If have dropped messages, use:
INFO: $ sudo journalctl --output=short --follow --all | sudo snappy-debug
kernel.printk_ratelimit = 0
= AppArmor =
Time: Dec  6 04:48:26
Log: apparmor="DENIED" operation="mknod" profile="snap.aws-iot-greengrass.greengrassd"
 name="/home/ubuntu/my-file.txt" pid=12345 comm="touch" requested_mask="c"
 denied_mask="c" fsuid=0 ouid=0
File: /home/ubuntu/my-file.txt (write)
Suggestion:
* add 'home' to 'plugs'
```

Contoh ini menunjukkan bahwa membuat `/home/ubuntu/my-file.txt` file menyebabkan kesalahan izin. Hal ini juga menunjukkan bahwa Anda menambahkan home ke `plugs`. Namun, saran ini tidak berlaku. Plug `home-for-greengrassd` dan `home-for-hooks` hanya diberikan akses hanya-baca.

Untuk informasi selengkapnya, lihat: snap snappy-debug di dokumentasi Snap.

error: tidak dapat melakukan tugas-tugas berikut: - Jalankan perintah layanan “start” untuk layanan ["greengrassd"] dari snap "aws-iot-greengrass" ([start snap. aws-iot-greengrass.greengrassd.service] gagal dengan status keluar 1: Job for snap. aws-iot-greengrass.greengrassd.service gagal karena proses kontrol keluar dengan kode kesalahan. Lihat “snap status systemctl. aws-iot-greengrass.greengrassd.service” dan “journalctl -xe” untuk detailnya.)

Solusi: Anda mungkin melihat kesalahan ini ketika `snap start aws-iot-greengrass` gagal untuk memulai AWS IoT Greengrass perangkat lunak Core.

Untuk informasi pemecahan masalah selengkapnya, jalankan perintah berikut:

```
sudo snap run aws-iot-greengrass.greengrassd
```

Contoh respons:

```
Couldn't find /snap/aws-iot-greengrass/44/greengrass/config/config.json.
```

Contoh ini menunjukkan bahwa AWS IoT Greengrass tidak dapat menemukan `config.json` file. Anda mungkin memeriksa file konfigurasi dan sertifikat.

/var/snap/ aws-iot-greengrass /current/ ggc-write-directory /packages/1.11.5/rootfs/merged bukan jalur absolut atau merupakan symlink.

Solusi: AWS IoT Greengrass snap hanya mendukung fungsi Lambda noncontainerized. Pastikan bahwa Anda menjalankan fungsi Lambda Anda dalam mode tanpa kontainer. Untuk informasi selengkapnya, lihat Pertimbangan ketika memilih fungsi Lambda kontainerisasi di AWS IoT Greengrass Version 1 Panduan Developer.

Daemon snapd gagal untuk me-restart setelah Anda menjalankan perintahsudo snap refresh snapd.

Solusi: Ikuti langkah 6 hingga 8 di Instal dan konfigurasi AWS IoT Greengrass snap untuk menambahkan sertifikat AWS IoT Greengrass dan konfigurasi file untuk snap AWS IoT Greengrass ini.

Arsip AWS IoT Greengrass instalasi perangkat lunak Core

Ketika Anda upgrade ke versi baru AWS IoT Greengrass perangkat lunak Core, Anda dapat mengarsip versi yang saat ini diinstal. Ini menjaga lingkungan instalasi Anda saat ini sehingga Anda dapat menguji versi perangkat lunak baru pada perangkat keras yang sama. Hal ini juga memudahkan untuk memutar kembali ke versi arsip Anda untuk alasan apa pun.

Untuk mengarsip instalasi saat ini dan menginstal versi baru

1. Mengunduh AWS IoT Greengrass perangkat lunak Core paket instalasi yang ingin Anda tingkatkan.
2. Salin paket ke perangkat core tujuan. Untuk petunjuk yang menunjukkan bagaimana transfer file, lihat langkah ini.

   > **Note**
   >
   > Anda menyalin sertifikat Anda saat ini, kunci, dan file konfigurasi untuk instalasi baru nanti.

   Jalankan perintah dalam langkah-langkah berikut di terminal perangkat core Anda.
3. Pastikan bahwa Greengrass daemon dihentikan pada perangkat core.

   a. Untuk memeriksa apakah daemon sedang berjalan:

   ```
   ps aux | grep -E 'greengrass.*daemon'
   ```

   Jika output berisi `root` entri untuk `/greengrass/ggc/packages/`*`ggc-version`*`/bin/daemon`, maka daemon sedang berjalan.

   > **Note**
   >
   > Prosedur ini ditulis dengan asumsi bahwa AWS IoT Greengrass perangkat lunak Core diinstal di `/greengrass` direktori.

   b. Untuk menghentikan daemon:

   ```
   cd /greengrass/ggc/core/
   ```

```
sudo ./greengrassd stop
```

4. Pindahkan direktori root Greengrass saat ini ke direktori yang berbeda.

```
sudo mv /greengrass /greengrass_backup
```

5. Untar perangkat lunak baru pada perangkat core. Ganti *arsitektur os* dan *versi* placeholder dalam perintah.

```
sudo tar -zxvf greengrass-os-architecture-version.tar.gz -C /
```

6. Salin sertifikat, kunci, dan file konfigurasi yang diarsipkan ke instalasi baru.

```
sudo cp /greengrass_backup/certs/* /greengrass/certs
sudo cp /greengrass_backup/config/* /greengrass/config
```

7. Mulai ulang daemon:

```
cd /greengrass/ggc/core/
sudo ./greengrassd start
```

Sekarang, Anda dapat membuat deployment kelompok untuk mengetes instalasi baru. Jika ada yang gagal, Anda dapat memulihkan instalasi yang diarsipkan.

Untuk memulihkan instalasi yang diarsipkan

1. Hentikan daemon.
2. Hapus direktori `/greengrass` baru.
3. Pindahkan `/greengrass_backup` direktori kembali ke `/greengrass`.
4. Mulai daemon.

Mengonfigurasi AWS IoT Greengrass core

Sebuah AWS IoT Greengrass core adalah AWS IoT hal (perangkat) yang bertindak sebagai hub atau gateway di lingkungan edge. Seperti perangkat AWS IoT lain, core yang ada di registri, memiliki perangkat bayangan, dan menggunakan sertifikat perangkat untuk mengotentikasi dengan AWS IoT Core dan AWS IoT Greengrass. Perangkat core menjalankan AWS IoT Greengrass perangkat

lunak Core, yang memungkinkannya untuk mengelola proses lokal untuk grup Greengrass, seperti komunikasi, sinkronisasi bayangan, dan pertukaran token.

Perangkat lunak core AWS IoT Greengrass menyediakan fungsi berikut:

- Deployment dan menjalankan lokal konektor dan fungsi Lambda.
- Memproses aliran data secara lokal dengan ekspor otomatis ke AWS Cloud.
- Olahpesan MQTT melalui jaringan lokal antara perangkat, konektor, dan fungsi Lambda menggunakan langganan terkelola.
- Olahpesan MQTT antara AWS IoT dan perangkat, konektor, dan fungsi Lambda dapat menggunakan langganan terkelola.
- Koneksi aman antara perangkat dan AWS Cloud menggunakan autentikasi dan otorisasi perangkat.
- Sinkronisasi bayangan lokal perangkat. Bayangan dapat dikonfigurasi untuk disinkronkan dengan AWS Cloud.
- Akses terkontrol ke perangkat lokal dan sumber daya volume.
- Deployment model machine learning dengan pelatihan cloud untuk menjalankan inferensi lokal.
- Deteksi alamat IP otomatis yang mengaktifkan perangkat untuk menemukan perangkat Core Greengrass.
- Deployment pusat dar konfigurasi grup yang baru atau diperbarui. Setelah data konfigurasi diunduh, perangkat core di-restart secara otomatis.
- Pembaruan perangkat lunak over-the-air (OTA) yang aman dari fungsi Lambda yang ditentukan pengguna.
- Aman, rahasia lokal yang terenkripsi dan akses yang dikendalikan oleh konektor dan fungsi Lambda.

AWS IoT Greengrass file konfigurasi core

File konfigurasi untuk AWS IoT Greengrass perangkat lunak Core adalah `config.json`. Ini terletak di `/greengrass-root/config` direktori.

> **Note**
>
> *greengrass-root* mewakili jalur di mana AWS IoT Greengrass perangkat lunak core diinstal pada perangkat Anda. Biasanya, ini adalah `/greengrass` direktori.

Jika Anda menggunakan opsi pembuatan Grup Default dari konsol AWS IoT Greengrass tersebut, maka file `config.json` di-deploy ke perangkat core dalam status bekerja.

Anda dapat meninjau isi file ini dengan menjalankan perintah berikut:

```
cat /greengrass-root/config/config.json
```

Berikut ini adalah contoh `config.json` file. Ini adalah versi yang dihasilkan ketika Anda membuat core dari AWS IoT Greengrass konsol.

GGC v1.11

```
{
    "coreThing": {
        "caPath": "root.ca.pem",
        "certPath": "hash.cert.pem",
        "keyPath": "hash.private.key",
        "thingArn": "arn:partition:iot:region:account-id:thing/core-thing-name",
        "iotHost": "host-prefix-ats.iot.region.amazonaws.com",
        "ggHost": "greengrass-ats.iot.region.amazonaws.com",
        "keepAlive": 600,
        "ggDaemonPort": 8000,
        "systemComponentAuthTimeout": 5000
    },
    "runtime": {
        "maxWorkItemCount": 1024,
        "maxConcurrentLimit": 25,
        "lruSize": 25,
        "mountAllBlockDevices": "no",
        "cgroup": {
            "useSystemd": "yes"
        }
    },
    "managedRespawn": false,
    "crypto": {
        "principals": {
            "SecretsManager": {
                "privateKeyPath": "file:///greengrass/certs/hash.private.key"
            },
            "IoTCertificate": {
                "privateKeyPath": "file:///greengrass/certs/hash.private.key",
```

```
                "certificatePath": "file:///greengrass/certs/hash.cert.pem"
            }
        },
        "caPath": "file:///greengrass/certs/root.ca.pem"
    },
    "writeDirectory": "/var/snap/aws-iot-greengrass/current/ggc-write-directory",
    "pidFileDirectory": "/var/snap/aws-iot-greengrass/current/pidFileDirectory"
}
```

File `config.json` mendukung properti berikut:

CoreThing

Bidang	Deskripsi	Catatan
`caPath`	Jalur ke AWS IoT root CA tergantung pada */greengrass-root* / `certs` direktori.	Untuk kompatibilitas backward dengan versi lebih awal dari 1.7.0. Properti ini diabaikan ketika `crypto` objek hadir. **Note** Pastikan bahwa titik akhir Anda sesuai dengan jenis sertifikat Anda.
`certPath`	Jalur ke sertifikat perangkat core relatif terhadap */greengrass-root* / `certs` direktori.	Untuk kompatibilitas backward dengan versi lebih awal dari 1.7.0. Properti ini diabaikan ketika `crypto` objek hadir.
`keyPath`	Jalur ke kunci privat core tergantung pada */greengrass-root* / `certs` direktori.	Untuk kompatibilitas backward dengan versi lebih awal dari 1.7.0. Properti ini

Bidang	Deskripsi	Catatan
		diabaikan ketika `crypto` objek hadir.
`thingArn`	Amazon Resource Name (ARN) dari AWS IoT hal yang mewakili AWS IoT Greengrass perangkat core.	Temukan ARN untuk core Anda di konsol AWS IoT Greengrass tersebut di bawah Cores, atau dengan menjalankan `aws greengrass get-core-definition-version` perintah CLI.

Bidang	Deskripsi	Catatan
`iotHost`	Titik akhir AWS IoT Anda.	Temukan titik akhir di konsol AWS IoT di bawah Pengaturan, atau dengan menjalankan `aws iot describe-endpoint --endpoint-type iot:Data-ATS` perintah CLI. Perintah ini mengembalikan titik akhir Amazon Trust Services (ATS). Untuk informasi lebih lanjut, lihat Dokumentasi autentikasi server. **Note** Pastikan bahwa titik akhir Anda sesuai dengan jenis sertifikat Anda. Pastikan bahwa titik akhir Anda sesuai dengan Wilayah AWS.

Bidang	Deskripsi	Catatan
`ggHost`	Titik akhir AWS IoT Greengrass Anda.	Ini adalah titik akhir `iotHost` Anda dengan prefiks host digantikan oleh greengrass (sebagai contoh, `greengrass-ats.iot.`*`region`*`.amazonaws.com`). Gunakan Wilayah AWS yang sama seperti `iotHost`. **Note** Pastikan bahwa titik akhir Anda sesuai dengan jenis sertifikat Anda. Pastikan bahwa titik akhir Anda sesuai dengan Wilayah AWS.
`iotMqttPort`	Opsional. Nomor port yang digunakan untuk komunikasi MQTT dengan AWS IoT.	Nilai yang valid adalah `8883` atau `443`. Nilai default-nya adalah `8883`. Untuk informasi selengkapnya, lihat Connect pada port 443 atau melalui proksi jaringan.
`iotHttpPort`	Opsional. Nomor port yang digunakan untuk membuat koneksi HTTPS ke AWS IoT.	Nilai yang valid adalah `8443` atau `443`. Nilai default-nya adalah `8443`. Untuk informasi selengkapnya, lihat Connect pada port 443 atau melalui proksi jaringan.

Bidang	Deskripsi	Catatan
`ggMqttPort`	Opsional. Nomor port yang digunakan untuk komunikasi MQTT melalui jaringan lokal.	Nilai yang valid adalah `1024` melalui `65535`. Nilai default-nya adalah `8883`. Untuk informasi selengkapnya, lihat the section called “Port MQTT untuk olahpesan lokal”.
`ggHttpPort`	Opsional. Nomor port yang digunakan untuk membuat koneksi HTTPS ke AWS IoT Greengrass layanan.	Nilai yang valid adalah `8443` atau `443`. Nilai default-nya adalah `8443`. Untuk informasi selengkapnya, lihat Connect pada port 443 atau melalui proksi jaringan.
`keepAlive`	Opsional. Periode `KeepAlive` MQTT, dalam detik.	Kisaran valid adalah antara 30 dan 1200 detik. Nilai default adalah `600`.
`networkProxy`	Opsional. Sebuah objek yang mendefinisikan server proksi untuk terhubung ke.	Server proksi dapat HTTP atau HTTPS. Untuk informasi selengkapnya, lihat Connect pada port 443 atau melalui proksi jaringan.
`mqttOperationTimeout`	Opsional. Jumlah waktu (dalam detik) untuk mengizinkan core Greengrass untuk menyelesaikan penerbitan, berlangganan, atau berhenti berlangganan operasi di koneksi MQTT ke AWS IoT Core.	Nilai bawaannya adalah 5. Nilai minimum adalah 5.

Bidang	Deskripsi	Catatan
`ggDaemonPort`	Opsional. Nomor port IPC core Greengrass.	Properti ini tersedia di AWS IoT Greengrass v1.11.0 atau lebih baru. Nilai yang valid adalah antara 1024 dan 65535. Nilai default adalah 8000.
`systemComponentAuthTimeout`	Opsional. Waktu (dalam milidetik) untuk mengizinkan core Greengrass IPC untuk menyelesaikan pengesahan.	Properti ini tersedia di AWS IoT Greengrass v1.11.0 atau lebih baru. Nilai yang valid adalah antara 500 dan 5000. Nilai default-nya adalah 5000.

runtime

Bidang	Deskripsi	Catatan
`maxWorkItemHitung`	Opsional. Jumlah maksimum item pekerjaan yang Greengrass daemon dapat memproses pada satu waktu. Item kerja yang melebihi batas ini diabaikan. Antrian item kerja dibagi oleh komponen sistem, fungsi Lambda yang ditetapkan pengguna, dan konektor.	Nilai default adalah 1024. Nilai maksimum dibatasi oleh perangkat keras perangkat Anda. Meningkatkan nilai ini meningkatkan memori yang AWS IoT Greengrass menggunakan. Anda dapat meningkatkan nilai ini jika Anda mengharapkan core Anda untuk menerima lalu lintas pesan MQTT berat.

Bidang	Deskripsi	Catatan
`maxConcurrentLimit`	Opsional. Jumlah maksimum pekerja Lambda yang tidak disematkan bersamaan yang dapat dimiliki oleh Greengrass daemon. Anda dapat menentukan integer yang berbeda untuk menimpa parameter ini.	Nilai default adalah 25. Nilai minimum ditetapkan oleh `lruSize`.
`lruSize`	Optional. Defines the minimum value for `maxConcurrentLimit` .	The default value is 25.
`mountAllBlockPerangkat`	Optional. Enables AWS IoT Greengrass to use bind mounts to mount all block devices into a container after setting up the OverlayFS.	Properti ini tersedia di AWS IoT Greengrass v1.11.0 atau lebih baru. Nilai yang valid adalah `yes` dan `no`. Nilai default-nya adalah `no`. Atur nilai ini ke `yes` jika direktori `/usr` Anda tidak di bawah `/` hierarki.
`postStartHealthCheckTimeout`	Optional. The time (in milliseconds) after starting that the Greengrass daemon waits for the health check to finish.	The default timeout is 30 seconds (30000 ms).
`cgroup`		

Bidang	Deskripsi	Catatan
`useSystemd`	Indicates whether your device uses `systemd`.	Valid values are `ya` or `tidak`. Run the `check_ggc_dependencies` script in Modul 1 to see if your device uses `systemd`.

kripto

Ini `crypto` Berisi properti yang mendukung penyimpanan kunci privat pada hardware security module (HSM) melalui PKCS #11 dan penyimpanan rahasia lokal. Lihat informasi selengkapnya di the section called “Keamanan utama”, the section called “Integrasi keamanan perangkat keras”, dan Men-deploy rahasia ke core. Konfigurasi untuk penyimpanan kunci privat pada HSMs atau di sistem file yang didukung.

Bidang	Deskripsi	Catatan
`caPath`	Jalur absolut ke CA root AWS IoT ini.	Harus berupa URI file dalam bentuk: `file:///absolute/path/to/file` . **Note** Pastikan bahwa titik akhir Anda sesuai dengan jenis sertifikat Anda.
`PKCS11`		
`OpenSSLEngine`	Opsional. Jalur absolut ke file `.so` mesin OpenSSL untuk mengaktifkan mendukung PKCS#11 pada OpenSSL.	Harus berupa jalur ke file pada sistem file. Properti ini diperlukan jika Anda menggunakan agen

Bidang	Deskripsi	Catatan
		pembaruan Greengrass OTA dengan keamanan perangkat keras. Untuk informasi selengkapnya, lihat the section called “Konfigurasi pembaruan OTA”.
`P11Provider`	Jalur absolut ke perpustakaan libdl-loadable implementasi PKCS#11.	Harus berupa jalur ke file pada sistem file.
`slotLabel`	Slot label yang digunakan untuk mengidentifikasi modul perangkat keras.	Harus sesuai dengan spesifikasi label PKCS#11.
`slotUserPin`	PIN pengguna yang digunakan untuk mengautentikasi core Greengrass ke modul.	Harus memiliki izin yang memadai untuk melakukan C_Sign dengan kunci privat yang dikonfigurasi.
`principals`		
`IoTCertificate`	The certificate and private key that the core uses to make requests to AWS IoT.	
`Sertifikat IoT. privateKeyPath`	Jalur ke kunci privat core.	Untuk penyimpanan sistem file, harus berupa URI file dalam bentuk: `file:///`*`absolute/path/to/file`* . Untuk penyimpanan HSM, harus jalur RFC 7512 PKCS#11 yang menentukan label objek.

Bidang	Deskripsi	Catatan
`IoTCertificate .certificatePath`	Jalur absolut untuk sertifikat perangkat core.	Harus berupa URI file dalam bentuk: `file:///`*`absolute/path/to/file`* .
`MQTT ServerCertificate`	Opsional. Kunci privat yang menggunakan core dalam kombinasi dengan sertifikat untuk bertindak sebagai server MQTT atau gateway.	
`MQTTServerCertificate . privateKeyPath`	Jalur ke kunci privat server MQTT lokal.	Gunakan nilai ini untuk menentukan kunci privat Anda sendiri untuk server MQTT lokal. Untuk penyimpanan sistem file, harus berupa URI file dalam bentuk: `file:///`*`absolute/path/to/file`* . Untuk penyimpanan HSM, harus jalur RFC 7512 PKCS#11 yang menentukan label objek. Jika properti ini dihilangkan, AWS IoT Greengrass memutar kunci berdasarkan pengaturan rotasi Anda. Jika ditentukan, pelanggan bertanggung jawab untuk memutar kunci.
`SecretsManager`	The private key that secures the data key used for encryption. For more information, see Men-deploy rahasia ke core.	

Bidang	Deskripsi	Catatan
`SecretsManager .privateKeyPath`	Jalur ke kunci privat secrets manager lokal.	Hanya kunci RSA yang didukung. Untuk penyimpanan sistem file, harus berupa URI file dalam bentuk: `file:///absolute/path/to/file` . Untuk penyimpanan HSM, harus jalur RFC 7512 PKCS#11 yang menentukan label objek. Kunci privat harus dibuat dengan menggunakan mekanisme padding PKCS #1 v1.5 ini.

Properti konfigurasi berikut juga didukung:

Bidang	Deskripsi	Catatan
`mqttMaxConnectionRetryInterval`	Opsional. Interval maksimum (dalam detik) antara koneksi retries MQTT jika koneksi terputus.	Tentukan nilai ini sebagai unsigned integer. Default-nya adalah `60`.
`managedRespawn`	Opsional. Mengindikasikan bahwa agen OTA perlu menjalankan kode kustom sebelum pembaruan.	Nilai yang valid adalah `true` atau `false`. Untuk informasi selengkapnya, lihat Perbarui OTA AWS IoT Greengrass perangkat lunak Core.
`writeDirectory`	Opsional. Direktori tulis di mana AWS IoT Greengras	Untuk informasi selengkapnya, lihat Mengonfigurasi

Bidang	Deskripsi	Catatan
	s membuat semua sumber daya baca/tulis.	direktori tulis untuk AWS IoT Greengrass.
pidFileDirectory	Opsional. AWS IoT Greengrass menyimpan process ID (PID) di bawah direktori ini.	Nilai default-nya adalah `/var/run`.

Extended life versions

Versi perangkat lunak AWS IoT Greengrass Core berikut ini berada dalam fase umur yang diperpanjang. Informasi ini disertakan untuk tujuan referensi saja.

GGC v1.10

```
{
  "coreThing" : {
    "caPath" : "root.ca.pem",
    "certPath" : "hash.cert.pem",
    "keyPath" : "hash.private.key",
    "thingArn" : "arn:partition:iot:region:account-id:thing/core-thing-name",
    "iotHost" : "host-prefix-ats.iot.region.amazonaws.com",
    "ggHost" : "greengrass-ats.iot.region.amazonaws.com",
    "keepAlive" : 600,
    "systemComponentAuthTimeout": 5000
  },
  "runtime" : {
    "maxWorkItemCount" : 1024,
    "maxConcurrentLimit" : 25,
    "lruSize": 25,
    "cgroup" : {
      "useSystemd" : "yes"
    }
  },
  "managedRespawn" : false,
  "crypto" : {
    "principals" : {
      "SecretsManager" : {
        "privateKeyPath" : "file:///greengrass/certs/hash.private.key"
```

```
      },
      "IoTCertificate" : {
        "privateKeyPath" : "file:///greengrass/certs/hash.private.key",
        "certificatePath" : "file:///greengrass/certs/hash.cert.pem"
      }
    },
    "caPath" : "file:///greengrass/certs/root.ca.pem"
  }
}
```

File `config.json` mendukung properti berikut:

CoreThing

Bidang	Deskripsi	Catatan
`caPath`	Jalur ke AWS IoT root CA tergantung pada */greengrass-root* / `certs` direktori.	Untuk kompatibilitas backward dengan versi lebih awal dari 1.7.0. Properti ini diabaikan ketika `crypto` objek hadir. **Note** Pastikan bahwa titik akhir Anda sesuai dengan jenis sertifikat Anda.
`certPath`	Jalur ke sertifikat perangkat core relatif terhadap */greengrass-root* / `certs` direktori.	Untuk kompatibilitas backward dengan versi lebih awal dari 1.7.0. Properti ini diabaikan ketika `crypto` objek hadir.
`keyPath`	Jalur ke kunci privat core tergantung pada */greengrass-root* / `certs` direktori.	Untuk kompatibilitas backward dengan versi lebih awal dari 1.7.0. Properti ini

Bidang	Deskripsi	Catatan
		diabaikan ketika `crypto` objek hadir.
`thingArn`	Amazon Resource Name (ARN) dari AWS IoT hal yang mewakili AWS IoT Greengrass perangkat core.	Temukan ARN untuk core Anda di konsol AWS IoT Greengrass tersebut di bawah Cores, atau dengan menjalankan `aws greengrass get-core-definition-version` perintah CLI.

Bidang	Deskripsi	Catatan
`iotHost`	Titik akhir AWS IoT Anda.	Temukan titik akhir di konsol AWS IoT di bawah Pengaturan, atau dengan menjalankan `aws iot describe-endpoint --endpoint-type iot:Data-ATS` perintah CLI. Perintah ini mengembalikan titik akhir Amazon Trust Services (ATS). Untuk informasi lebih lanjut, lihat Dokumentasi autentikasi server. **Note** Pastikan bahwa titik akhir Anda sesuai dengan jenis sertifikat Anda. Pastikan bahwa titik akhir Anda sesuai dengan Wilayah AWS.

Bidang	Deskripsi	Catatan
`ggHost`	Titik akhir AWS IoT Greengrass Anda.	Ini adalah titik akhir `iotHost` Anda dengan prefiks host digantikan oleh greengrass (sebagai contoh, `greengrass-ats.iot.` *`region`*`.amazonaws.com`). Gunakan Wilayah AWS yang sama seperti `iotHost`. **Note** Pastikan bahwa titik akhir Anda sesuai dengan jenis sertifikat Anda. Pastikan bahwa titik akhir Anda sesuai dengan Wilayah AWS.
`iotMqttPort`	Opsional. Nomor port yang digunakan untuk komunikasi MQTT dengan AWS IoT.	Nilai yang valid adalah `8883` atau `443`. Nilai default-nya adalah `8883`. Untuk informasi selengkapnya, lihat Connect pada port 443 atau melalui proksi jaringan.
`iotHttpPort`	Opsional. Nomor port yang digunakan untuk membuat koneksi HTTPS ke AWS IoT.	Nilai yang valid adalah `8443` atau `443`. Nilai default-nya adalah `8443`. Untuk informasi selengkapnya, lihat Connect pada port 443 atau melalui proksi jaringan.

Bidang	Deskripsi	Catatan
`ggMqttPort`	Opsional. Nomor port yang digunakan untuk komunikasi MQTT melalui jaringan lokal.	Nilai yang valid adalah `1024` melalui `65535`. Nilai default-nya adalah `8883`. Untuk informasi selengkapnya, lihat the section called “Port MQTT untuk olahpesan lokal”.
`ggHttpPort`	Opsional. Nomor port yang digunakan untuk membuat koneksi HTTPS ke AWS IoT Greengrass layanan.	Nilai yang valid adalah `8443` atau `443`. Nilai default-nya adalah `8443`. Untuk informasi selengkapnya, lihat Connect pada port 443 atau melalui proksi jaringan.
`keepAlive`	Opsional. Periode `KeepAlive` MQTT, dalam detik.	Kisaran valid adalah antara 30 dan 1200 detik. Nilai default adalah `600`.
`networkProxy`	Opsional. Sebuah objek yang mendefinisikan server proksi untuk terhubung ke.	Server proksi dapat HTTP atau HTTPS. Untuk informasi selengkapnya, lihat Connect pada port 443 atau melalui proksi jaringan.
`mqttOperationTimeout`	Opsional. Jumlah waktu (dalam detik) untuk mengizinkan core Greengrass untuk menyelesaikan penerbitan, berlangganan, atau berhenti berlangganan operasi di koneksi MQTT ke AWS IoT Core.	Properti ini tersedia mulai di AWS IoT Greengrass v1.10.2. Nilai bawaannya adalah 5. Nilai minimum adalah 5.

runtime

Bidang	Deskripsi	Catatan
`maxWorkItemHitung`	Opsional. Jumlah maksimum item pekerjaan yang Greengrass daemon dapat memproses pada satu waktu. Item kerja yang melebihi batas ini diabaikan. Antrian item kerja dibagi oleh komponen sistem, fungsi Lambda yang ditetapkan pengguna, dan konektor.	Nilai default adalah 1024. Nilai maksimum dibatasi oleh perangkat keras perangkat Anda. Meningkatkan nilai ini meningkatkan memori yang AWS IoT Greengrass menggunakan. Anda dapat meningkatkan nilai ini jika Anda mengharapkan core Anda untuk menerima lalu lintas pesan MQTT berat.
`maxConcurrentLimit`	Opsional. Jumlah maksimum pekerja Lambda yang tidak disematkan bersamaan yang dapat dimiliki oleh Greengras s daemon. Anda dapat menentukan integer yang berbeda untuk menimpa parameter ini.	Nilai default adalah 25. Nilai minimum ditetapkan oleh `lruSize`.
`lruSize`	Optional. Defines the minimum value for `maxConcurrentLimit` .	The default value is 25.
`postStartHealthChe ckTimeout`	Optional. The time (in milliseconds) after starting that the Greengrass daemon waits for the health check to finish.	The default timeout is 30 seconds (30000 ms).

Bidang	Deskripsi	Catatan
`cgroup`		
`useSystemd`	Indicates whether your device uses `systemd`.	Valid values are `ya` or `tidak`. Run the `check_ggc_dependencies` script in Modul 1 to see if your device uses `systemd`.

kripto

Ini `crypto` Berisi properti yang mendukung penyimpanan kunci privat pada hardware security module (HSM) melalui PKCS #11 dan penyimpanan rahasia lokal. Lihat informasi selengkapnya di the section called “Keamanan utama”, the section called “Integrasi keamanan perangkat keras”, dan Men-deploy rahasia ke core. Konfigurasi untuk penyimpanan kunci privat pada HSMs atau di sistem file yang didukung.

Bidang	Deskripsi	Catatan
`caPath`	Jalur absolut ke CA root AWS IoT ini.	Harus berupa URI file dalam bentuk: `file:///absolute/path/to/file` . **Note** Pastikan bahwa titik akhir Anda sesuai dengan jenis sertifikat Anda.
`PKCS11`		
`OpenSSLEngine`	Opsional. Jalur absolut ke file `.so` mesin OpenSSL	Harus berupa jalur ke file pada sistem file.

Bidang	Deskripsi	Catatan
	untuk mengaktifkan mendukung PKCS#11 pada OpenSSL.	Properti ini diperlukan jika Anda menggunakan agen pembaruan Greengrass OTA dengan keamanan perangkat keras. Untuk informasi selengkapnya, lihat the section called “Konfigurasi pembaruan OTA”.
`P11Provider`	Jalur absolut ke perpustak aan libdl-loadable implement asi PKCS#11.	Harus berupa jalur ke file pada sistem file.
`slotLabel`	Slot label yang digunakan untuk mengidentifikasi modul perangkat keras.	Harus sesuai dengan spesifikasi label PKCS#11.
`slotUserPin`	PIN pengguna yang digunakan untuk mengauten tikasi core Greengrass ke modul.	Harus memiliki izin yang memadai untuk melakukan C_Sign dengan kunci privat yang dikonfigurasi.
`principals`		
`IoTCertificate`	The certificate and private key that the core uses to make requests to AWS IoT.	

Bidang	Deskripsi	Catatan
`Sertifikat IoT. privateKeyPath`	Jalur ke kunci privat core.	Untuk penyimpanan sistem file, harus berupa URI file dalam bentuk: `file:///absolute/path/to/file` . Untuk penyimpanan HSM, harus jalur RFC 7512 PKCS#11 yang menentukan label objek.
`IoTCertificate .certificatePath`	Jalur absolut untuk sertifikat perangkat core.	Harus berupa URI file dalam bentuk: `file:///absolute/path/to/file` .
`MQTT ServerCer tificate`	Opsional. Kunci privat yang menggunakan core dalam kombinasi dengan sertifikat untuk bertindak sebagai server MQTT atau gateway.	

Bidang	Deskripsi	Catatan
`MQTTServerCertificate . privateKeyPath`	Jalur ke kunci privat server MQTT lokal.	Gunakan nilai ini untuk menentukan kunci privat Anda sendiri untuk server MQTT lokal. Untuk penyimpanan sistem file, harus berupa URI file dalam bentuk: `file:///`*`absolute/path/to/file`* . Untuk penyimpanan HSM, harus jalur RFC 7512 PKCS#11 yang menentukan label objek. Jika properti ini dihilangkan, AWS IoT Greengrass memutar kunci berdasarkan pengaturan rotasi Anda. Jika ditentukan, pelanggan bertanggung jawab untuk memutar kunci.
`SecretsManager`	The private key that secures the data key used for encryption. For more information, see Men-deploy rahasia ke core.	

Bidang	Deskripsi	Catatan
`SecretsManager .privateKeyPath`	Jalur ke kunci privat secrets manager lokal.	Hanya kunci RSA yang didukung. Untuk penyimpanan sistem file, harus berupa URI file dalam bentuk: `file:///`*`absolute/path/to/file`* . Untuk penyimpanan HSM, harus jalur RFC 7512 PKCS#11 yang menentukan label objek. Kunci privat harus dibuat dengan menggunakan mekanisme padding PKCS #1 v1.5 ini.

Properti konfigurasi berikut juga didukung:

Bidang	Deskripsi	Catatan
`mqttMaxConnectionRetryInterval`	Opsional. Interval maksimum (dalam detik) antara koneksi retries MQTT jika koneksi terputus.	Tentukan nilai ini sebagai unsigned integer. Default-nya adalah `60`.
`managedRespawn`	Opsional. Mengindikasikan bahwa agen OTA perlu menjalankan kode kustom sebelum pembaruan.	Nilai yang valid adalah `true` atau `false`. Untuk informasi selengkapnya, lihat Perbarui OTA AWS IoT Greengrass perangkat lunak Core.
`writeDirectory`	Opsional. Direktori tulis di mana AWS IoT Greengras	Untuk informasi selengkapnya, lihat Mengonfigurasi

Bidang	Deskripsi	Catatan
	s membuat semua sumber daya baca/tulis.	direktori tulis untuk AWS IoT Greengrass.

GGC v1.9

```
{
  "coreThing" : {
    "caPath" : "root.ca.pem",
    "certPath" : "hash.cert.pem",
    "keyPath" : "hash.private.key",
    "thingArn" : "arn:partition:iot:region:account-id:thing/core-thing-name",
    "iotHost" : "host-prefix-ats.iot.region.amazonaws.com",
    "ggHost" : "greengrass-ats.iot.region.amazonaws.com",
    "keepAlive" : 600
  },
  "runtime" : {
    "cgroup" : {
      "useSystemd" : "yes"
    }
  },
  "managedRespawn" : false,
  "crypto" : {
    "principals" : {
      "SecretsManager" : {
        "privateKeyPath" : "file:///greengrass/certs/hash.private.key"
      },
      "IoTCertificate" : {
        "privateKeyPath" : "file:///greengrass/certs/hash.private.key",
        "certificatePath" : "file:///greengrass/certs/hash.cert.pem"
      }
    },
    "caPath" : "file:///greengrass/certs/root.ca.pem"
  }
}
```

File `config.json` mendukung properti berikut:

CoreThing

Bidang	Deskripsi	Catatan
`caPath`	Jalur ke AWS IoT root CA tergantung pada `/greengrass-root /certs` direktori.	Untuk kompatibilitas backward dengan versi lebih awal dari 1.7.0. Properti ini diabaikan ketika `crypto` objek hadir. **Note** Pastikan bahwa titik akhir Anda sesuai dengan jenis sertifikat Anda.
`certPath`	Jalur ke sertifikat perangkat core relatif terhadap `/greengrass-root /certs` direktori.	Untuk kompatibilitas backward dengan versi lebih awal dari 1.7.0. Properti ini diabaikan ketika `crypto` objek hadir.
`keyPath`	Jalur ke kunci privat core tergantung pada `/greengrass-root /certs` direktori.	Untuk kompatibilitas backward dengan versi lebih awal dari 1.7.0. Properti ini diabaikan ketika `crypto` objek hadir.
`thingArn`	Amazon Resource Name (ARN) dari AWS IoT hal yang mewakili AWS IoT Greengrass perangkat core.	Temukan ARN untuk core Anda di konsol AWS IoT Greengrass tersebut di bawah Cores, atau dengan menjalankan `aws greengrass get-core-definition-version` perintah CLI.

Bidang	Deskripsi	Catatan
`iotHost`	Titik akhir AWS IoT Anda.	Temukan titik akhir di konsol AWS IoT di bawah Pengaturan, atau dengan menjalankan `aws iot describe-endpoint --endpoint-type iot:Data-ATS` perintah CLI. Perintah ini mengembalikan titik akhir Amazon Trust Services (ATS). Untuk informasi lebih lanjut, lihat Dokumentasi autentikasi server. **Note** Pastikan bahwa titik akhir Anda sesuai dengan jenis sertifikat Anda. Pastikan bahwa titik akhir Anda sesuai dengan Wilayah AWS.

Bidang	Deskripsi	Catatan
`ggHost`	Titik akhir AWS IoT Greengrass Anda.	Ini adalah titik akhir `iotHost` Anda dengan prefiks host digantikan oleh greengrass (sebagai contoh, `greengrass-ats.iot.`*`region`*`.amazonaws.com`). Gunakan Wilayah AWS yang sama seperti `iotHost`. **Note** Pastikan bahwa titik akhir Anda sesuai dengan jenis sertifikat Anda. Pastikan bahwa titik akhir Anda sesuai dengan Wilayah AWS.
`iotMqttPort`	Opsional. Nomor port yang digunakan untuk komunikasi MQTT dengan AWS IoT.	Nilai yang valid adalah `8883` atau `443`. Nilai default-nya adalah `8883`. Untuk informasi selengkapnya, lihat Connect pada port 443 atau melalui proksi jaringan.
`iotHttpPort`	Opsional. Nomor port yang digunakan untuk membuat koneksi HTTPS ke AWS IoT.	Nilai yang valid adalah `8443` atau `443`. Nilai default-nya adalah `8443`. Untuk informasi selengkapnya, lihat Connect pada port 443 atau melalui proksi jaringan.

Bidang	Deskripsi	Catatan
`ggHttpPort`	Opsional. Nomor port yang digunakan untuk membuat koneksi HTTPS ke AWS IoT Greengrass layanan.	Nilai yang valid adalah `8443` atau `443`. Nilai default-n ya adalah `8443`. Untuk informasi selengkapnya, lihat Connect pada port 443 atau melalui proksi jaringan.
`keepAlive`	Opsional. Periode `KeepAlive` MQTT, dalam detik.	Kisaran valid adalah antara 30 dan 1200 detik. Nilai default adalah `600`.
`networkProxy`	Opsional. Sebuah objek yang mendefinisikan server proksi untuk terhubung ke.	Server proksi dapat HTTP atau HTTPS. Untuk informasi selengkapnya, lihat Connect pada port 443 atau melalui proksi jaringan.

runtime

Bidang	Deskripsi	Catatan
`maxConcurrentLimit`	Opsional. Jumlah maksimum pekerja Lambda yang tidak disematkan bersamaan yang dapat dimiliki oleh Greengras s daemon. Anda dapat menentukan integer yang berbeda untuk menimpa parameter ini.	Nilai default adalah 25. Nilai minimum ditetapkan oleh `lruSize`.
`lruSize`	Optional. Defines the minimum value for `maxConcurrentLimit` .	The default value is 25.

Bidang	Deskripsi	Catatan
`postStartHealthCheckTimeout`	Optional. The time (in milliseconds) after starting that the Greengrass daemon waits for the health check to finish.	The default timeout is 30 seconds (30000 ms).
`cgroup`		
`useSystemd`	Indicates whether your device uses `systemd`.	Valid values are `ya` or `tidak`. Run the `check_ggc_dependencies` script in Modul 1 to see if your device uses `systemd`.

kripto

Objek `crypto` ditambahkan di v1.7.0. Ini memperkenalkan properti yang mendukung penyimpanan kunci privat pada hardware security module (HSM) melalui PKCS #11 dan penyimpanan rahasia lokal. Lihat informasi selengkapnya di the section called “Keamanan utama”, the section called “Integrasi keamanan perangkat keras”, dan Men-deploy rahasia ke core. Konfigurasi untuk penyimpanan kunci privat pada HSMs atau di sistem file yang didukung.

Bidang	Deskripsi	Catatan
`caPath`	Jalur absolut ke CA root AWS IoT ini.	Harus berupa URI file dalam bentuk: `file:///absolute/path/to/file` . **Note** Pastikan bahwa titik akhir Anda

Bidang	Deskripsi	Catatan
		sesuai dengan jenis sertifikat Anda.
`PKCS11`		
`OpenSSLEngine`	Opsional. Jalur absolut ke file `.so` mesin OpenSSL untuk mengaktifkan mendukung PKCS#11 pada OpenSSL.	Harus berupa jalur ke file pada sistem file. Properti ini diperlukan jika Anda menggunakan agen pembaruan Greengrass OTA dengan keamanan perangkat keras. Untuk informasi selengkapnya, lihat the section called "Konfigurasi pembaruan OTA".
`P11Provider`	Jalur absolut ke perpustak aan libdl-loadable implement asi PKCS#11.	Harus berupa jalur ke file pada sistem file.
`slotLabel`	Slot label yang digunakan untuk mengidentifikasi modul perangkat keras.	Harus sesuai dengan spesifikasi label PKCS#11.
`slotUserPin`	PIN pengguna yang digunakan untuk mengauten tikasi core Greengrass ke modul.	Harus memiliki izin yang memadai untuk melakukan C_Sign dengan kunci privat yang dikonfigurasi.
`principals`		
`IoTCertificate`	The certificate and private key that the core uses to make requests to AWS IoT.	

Bidang	Deskripsi	Catatan
`Sertifikat IoT. privateKeyPath`	Jalur ke kunci privat core.	Untuk penyimpanan sistem file, harus berupa URI file dalam bentuk: `file:///absolute/path/to/file` . Untuk penyimpanan HSM, harus jalur RFC 7512 PKCS#11 yang menentukan label objek.
`IoTCertificate .certificatePath`	Jalur absolut untuk sertifikat perangkat core.	Harus berupa URI file dalam bentuk: `file:///absolute/path/to/file` .
`MQTT ServerCer tificate`	Opsional. Kunci privat yang menggunakan core dalam kombinasi dengan sertifikat untuk bertindak sebagai server MQTT atau gateway.	

Bidang	Deskripsi	Catatan
`MQTTServerCertificate . privateKeyPath`	Jalur ke kunci privat server MQTT lokal.	Gunakan nilai ini untuk menentukan kunci privat Anda sendiri untuk server MQTT lokal. Untuk penyimpanan sistem file, harus berupa URI file dalam bentuk: `file:///`*`absolute/path/to/file`* . Untuk penyimpanan HSM, harus jalur RFC 7512 PKCS#11 yang menentukan label objek. Jika properti ini dihilangkan, AWS IoT Greengrass memutar kunci berdasarkan pengaturan rotasi Anda. Jika ditentukan, pelanggan bertanggung jawab untuk memutar kunci.
`SecretsManager`	The private key that secures the data key used for encryption. For more information, see Men-deploy rahasia ke core.	

Bidang	Deskripsi	Catatan
`SecretsManager .privateKeyPath`	Jalur ke kunci privat secrets manager lokal.	Hanya kunci RSA yang didukung. Untuk penyimpanan sistem file, harus berupa URI file dalam bentuk: `file:///`*`absolute/path/to/file`* . Untuk penyimpanan HSM, harus jalur RFC 7512 PKCS#11 yang menentukan label objek. Kunci privat harus dibuat dengan menggunakan mekanisme padding PKCS #1 v1.5 ini.

Properti konfigurasi berikut juga didukung.

Bidang	Deskripsi	Catatan
`mqttMaxConnectionRetryInterval`	Opsional. Interval maksimum (dalam detik) antara koneksi retries MQTT jika koneksi terputus.	Tentukan nilai ini sebagai unsigned integer. Default-nya adalah `60`.
`managedRespawn`	Opsional. Mengindikasikan bahwa agen OTA perlu menjalankan kode kustom sebelum pembaruan.	Nilai yang valid adalah `true` atau `false`. Untuk informasi selengkapnya, lihat Perbarui OTA AWS IoT Greengrass perangkat lunak Core.
`writeDirectory`	Opsional. Direktori tulis di mana AWS IoT Greengras	Untuk informasi selengkapnya, lihat Mengonfigurasi

Bidang	Deskripsi	Catatan
	s membuat semua sumber daya baca/tulis.	direktori tulis untuk AWS IoT Greengrass.

GGC v1.8

```
{
  "coreThing" : {
    "caPath" : "root.ca.pem",
    "certPath" : "hash.cert.pem",
    "keyPath" : "hash.private.key",
    "thingArn" : "arn:aws:iot:region:account-id:thing/core-thing-name",
    "iotHost" : "host-prefix-ats.iot.region.amazonaws.com",
    "ggHost" : "greengrass-ats.iot.region.amazonaws.com",
    "keepAlive" : 600
  },
  "runtime" : {
    "cgroup" : {
      "useSystemd" : "yes"
    }
  },
  "managedRespawn" : false,
  "crypto" : {
    "principals" : {
      "SecretsManager" : {
        "privateKeyPath" : "file:///greengrass/certs/hash.private.key"
      },
      "IoTCertificate" : {
        "privateKeyPath" : "file:///greengrass/certs/hash.private.key",
        "certificatePath" : "file:///greengrass/certs/hash.cert.pem"
      }
    },
    "caPath" : "file:///greengrass/certs/root.ca.pem"
  }
}
```

File `config.json` mendukung properti berikut.

CoreThing

Bidang	Deskripsi	Catatan
`caPath`	Jalur ke AWS IoT root CA tergantung pada */greengrass-root* / `certs` direktori.	Untuk kompatibilitas backward dengan versi lebih awal dari 1.7.0. Properti ini diabaikan ketika `crypto` objek hadir. **Note** Pastikan bahwa titik akhir Anda sesuai dengan jenis sertifikat Anda.
`certPath`	Jalur ke sertifikat perangkat core relatif terhadap */greengrass-root* / `certs` direktori.	Untuk kompatibilitas backward dengan versi lebih awal dari 1.7.0. Properti ini diabaikan ketika `crypto` objek hadir.
`keyPath`	Jalur ke kunci privat core tergantung pada */greengrass-root* / `certs` direktori.	Untuk kompatibilitas backward dengan versi lebih awal dari 1.7.0. Properti ini diabaikan ketika `crypto` objek hadir.
`thingArn`	Amazon Resource Name (ARN) dari AWS IoT hal yang mewakili AWS IoT Greengrass perangkat core.	Temukan ARN untuk core Anda di konsol AWS IoT Greengrass tersebut di bawah Cores, atau dengan menjalankan `aws greengrass get-core-definition-version` perintah CLI.

Bidang	Deskripsi	Catatan
`iotHost`	Titik akhir AWS IoT Anda.	Temukan titik akhir di konsol AWS IoT di bawah Pengaturan, atau dengan menjalankan `aws iot describe-endpoint --endpoint-type iot:Data-ATS` perintah CLI. Perintah ini mengembalikan titik akhir Amazon Trust Services (ATS). Untuk informasi lebih lanjut, lihat Dokumentasi autentikasi server. **Note** Pastikan bahwa titik akhir Anda sesuai dengan jenis sertifikat Anda. Pastikan titik akhir Anda sesuai dengan Wilayah AWS.

Bidang	Deskripsi	Catatan
`ggHost`	Titik akhir AWS IoT Greengrass Anda.	Ini adalah titik akhir `iotHost` Anda dengan prefiks host digantikan oleh greengrass (sebagai contoh, `greengrass-ats.iot.`*`region`*`.amazonaws.com`). Gunakan Wilayah AWS yang sama seperti `iotHost`. **Note** Pastikan bahwa titik akhir Anda sesuai dengan jenis sertifikat Anda. Pastikan titik akhir Anda sesuai dengan Wilayah AWS.
`iotMqttPort`	Opsional. Nomor port yang digunakan untuk komunikasi MQTT dengan AWS IoT.	Nilai yang valid adalah `8883` atau `443`. Nilai default-nya adalah `8883`. Untuk informasi selengkapnya, lihat Connect pada port 443 atau melalui proksi jaringan.
`iotHttpPort`	Opsional. Nomor port yang digunakan untuk membuat koneksi HTTPS ke AWS IoT.	Nilai yang valid adalah `8443` atau `443`. Nilai default-nya adalah `8443`. Untuk informasi selengkapnya, lihat Connect pada port 443 atau melalui proksi jaringan.

Bidang	Deskripsi	Catatan
`ggHttpPort`	Opsional. Nomor port yang digunakan untuk membuat koneksi HTTPS ke AWS IoT Greengrass layanan.	Nilai yang valid adalah `8443` atau `443`. Nilai default-n ya adalah `8443`. Untuk informasi selengkapnya, lihat Connect pada port 443 atau melalui proksi jaringan.
`keepAlive`	Opsional. Periode `KeepAlive` MQTT, dalam detik.	Kisaran valid adalah antara 30 dan 1200 detik. Nilai default adalah `600`.
`networkProxy`	Opsional. Sebuah objek yang mendefinisikan server proksi untuk terhubung ke.	Server proksi dapat HTTP atau HTTPS. Untuk informasi selengkapnya, lihat Connect pada port 443 atau melalui proksi jaringan.

runtime

Bidang	Deskripsi	Catatan
`cgroup`		
`useSystemd`	Indicates whether your device uses `systemd`.	Valid values are `ya` or `tidak`. Run the `check_ggc_dependencies` script in Modul 1 to see if your device uses `systemd`.

kripto

Objek `crypto` ditambahkan di v1.7.0. Ini memperkenalkan properti yang mendukung penyimpanan kunci privat pada hardware security module (HSM) melalui PKCS #11 dan penyimpanan rahasia lokal. Lihat informasi selengkapnya di the section called “Keamanan

utama", the section called "Integrasi keamanan perangkat keras", dan Men-deploy rahasia ke core. Konfigurasi untuk penyimpanan kunci privat pada HSMs atau di sistem file yang didukung.

Bidang	Deskripsi	Catatan
`caPath`	Jalur absolut ke CA root AWS IoT ini.	Harus berupa URI file dalam bentuk: `file:///absolute/path/to/file` . **Note** Pastikan bahwa titik akhir Anda sesuai dengan jenis sertifikat Anda.
`PKCS11`		
`OpenSSLEngine`	Opsional. Jalur absolut ke file `.so` mesin OpenSSL untuk mengaktifkan mendukung PKCS#11 pada OpenSSL.	Harus berupa jalur ke file pada sistem file. Properti ini diperlukan jika Anda menggunakan agen pembaruan Greengrass OTA dengan keamanan perangkat keras. Untuk informasi selengkapnya, lihat the section called "Konfigurasi pembaruan OTA".
`P11Provider`	Jalur absolut ke perpustakaan libdl-loadable implementasi PKCS#11.	Harus berupa jalur ke file pada sistem file.

Bidang	Deskripsi	Catatan
`slotLabel`	Slot label yang digunakan untuk mengidentifikasi modul perangkat keras.	Harus sesuai dengan spesifikasi label PKCS#11.
`slotUserPin`	PIN pengguna yang digunakan untuk mengautentikasi core Greengrass ke modul.	Harus memiliki izin yang memadai untuk melakukan C_Sign dengan kunci privat yang dikonfigurasi.
`principals`		
`IoTCertificate`	The certificate and private key that the core uses to make requests to AWS IoT.	
`Sertifikat IoT. privateKeyPath`	Jalur ke kunci privat core.	Untuk penyimpanan sistem file, harus berupa URI file dalam bentuk: `file:///`*`absolute/path/to/file`* . Untuk penyimpanan HSM, harus jalur RFC 7512 PKCS#11 yang menentukan label objek.
`IoTCertificate .certificatePath`	Jalur absolut untuk sertifikat perangkat core.	Harus berupa URI file dalam bentuk: `file:///`*`absolute/path/to/file`* .
`MQTT ServerCertificate`	Opsional. Kunci privat yang menggunakan core dalam kombinasi dengan sertifikat untuk bertindak sebagai server MQTT atau gateway.	

Bidang	Deskripsi	Catatan
`MQTTServerCertificate . privateKeyPath`	Jalur ke kunci privat server MQTT lokal.	Gunakan nilai ini untuk menentukan kunci privat Anda sendiri untuk server MQTT lokal. Untuk penyimpanan sistem file, harus berupa URI file dalam bentuk: `file:///`*`absolute/path/to/file`* . Untuk penyimpanan HSM, harus jalur RFC 7512 PKCS#11 yang menentukan label objek. Jika properti ini dihilangkan, AWS IoT Greengrass memutar kunci berdasarkan pengaturan rotasi Anda. Jika ditentukan, pelanggan bertanggung jawab untuk memutar kunci.
`SecretsManager`	The private key that secures the data key used for encryption. For more information, see Men-deploy rahasia ke core.	

Bidang	Deskripsi	Catatan
`SecretsManager .privateKeyPath`	Jalur ke kunci privat secrets manager lokal.	Hanya kunci RSA yang didukung. Untuk penyimpanan sistem file, harus berupa URI file dalam bentuk: `file:///`*`absolute/path/to/file`* . Untuk penyimpanan HSM, harus jalur RFC 7512 PKCS#11 yang menentukan label objek. Kunci privat harus dibuat dengan menggunakan mekanisme padding PKCS #1 v1.5 ini.

Properti konfigurasi berikut juga didukung:

Bidang	Deskripsi	Catatan
`mqttMaxConnectionRetryInterval`	Opsional. Interval maksimum (dalam detik) antara koneksi retries MQTT jika koneksi terputus.	Tentukan nilai ini sebagai unsigned integer. Default-nya adalah `60`.
`managedRespawn`	Opsional. Mengindikasikan bahwa agen OTA perlu menjalankan kode kustom sebelum pembaruan.	Nilai yang valid adalah `true` atau `false`. Untuk informasi selengkapnya, lihat Perbarui OTA AWS IoT Greengrass perangkat lunak Core.
`writeDirectory`	Opsional. Direktori tulis di mana AWS IoT Greengras	Untuk informasi selengkapnya, lihat Mengonfigurasi

Bidang	Deskripsi	Catatan
	s membuat semua sumber daya baca/tulis.	direktori tulis untuk AWS IoT Greengrass.

GGC v1.7

```
{
  "coreThing" : {
    "caPath" : "root.ca.pem",
    "certPath" : "hash.cert.pem",
    "keyPath" : "hash.private.key",
    "thingArn" : "arn:aws:iot:region:account-id:thing/core-thing-name",
    "iotHost" : "host-prefix-ats.iot.region.amazonaws.com",
    "ggHost" : "greengrass-ats.iot.region.amazonaws.com",
    "keepAlive" : 600
  },
  "runtime" : {
    "cgroup" : {
      "useSystemd" : "yes"
    }
  },
  "managedRespawn" : false,
  "crypto" : {
    "principals" : {
      "SecretsManager" : {
        "privateKeyPath" : "file:///greengrass/certs/hash.private.key"
      },
      "IoTCertificate" : {
        "privateKeyPath" : "file:///greengrass/certs/hash.private.key",
        "certificatePath" : "file:///greengrass/certs/hash.cert.pem"
      }
    },
    "caPath" : "file:///greengrass/certs/root.ca.pem"
  }
}
```

File `config.json` mendukung properti berikut:

CoreThing

Bidang	Deskripsi	Catatan
`caPath`	Jalur ke AWS IoT root CA tergantung pada */greengrass-root* `/certs` direktori.	Untuk kompatibilitas backward dengan versi lebih awal dari 1.7.0. Properti ini diabaikan ketika `crypto` objek hadir. **Note** Pastikan bahwa titik akhir Anda sesuai dengan jenis sertifikat Anda.
`certPath`	Jalur ke sertifikat perangkat core relatif terhadap */greengrass-root* `/certs` direktori.	Untuk kompatibilitas backward dengan versi lebih awal dari 1.7.0. Properti ini diabaikan ketika `crypto` objek hadir.
`keyPath`	Jalur ke kunci privat core tergantung pada */greengrass-root* `/certs` direktori.	Untuk kompatibilitas backward dengan versi lebih awal dari 1.7.0. Properti ini diabaikan ketika `crypto` objek hadir.
`thingArn`	Amazon Resource Name (ARN) dari AWS IoT hal yang mewakili AWS IoT Greengrass perangkat core.	Temukan ARN untuk core Anda di konsol AWS IoT Greengrass tersebut di bawah Cores, atau dengan menjalankan `aws greengrass get-core-definition-version` perintah CLI.

Bidang	Deskripsi	Catatan
`iotHost`	Titik akhir AWS IoT Anda.	Temukan titik akhir di konsol AWS IoT di bawah Pengaturan, atau dengan menjalankan `aws iot describe-endpoint --endpoint-type iot:Data-ATS` perintah CLI. Perintah ini mengembalikan titik akhir Amazon Trust Services (ATS). Untuk informasi lebih lanjut, lihat Dokumentasi autentikasi server. **Note** Pastikan bahwa titik akhir Anda sesuai dengan jenis sertifikat Anda. Pastikan titik akhir Anda sesuai dengan Wilayah AWS.

Bidang	Deskripsi	Catatan
`ggHost`	Titik akhir AWS IoT Greengrass Anda.	Ini adalah titik akhir `iotHost` Anda dengan prefiks host digantikan oleh greengrass (sebagai contoh, `greengrass-ats.iot.` *`region`*`.amazonaws.com`). Gunakan Wilayah AWS yang sama seperti `iotHost`. **Note** Pastikan bahwa titik akhir Anda sesuai dengan jenis sertifikat Anda. Pastikan titik akhir Anda sesuai dengan Wilayah AWS.
`iotMqttPort`	Opsional. Nomor port yang digunakan untuk komunikasi MQTT dengan AWS IoT.	Nilai yang valid adalah 8883 atau 443. Nilai default-nya adalah 8883. Untuk informasi selengkapnya, lihat Connect pada port 443 atau melalui proksi jaringan.
`keepAlive`	Opsional. Periode `KeepAlive` MQTT, dalam detik.	Kisaran valid adalah antara 30 dan 1200 detik. Nilai default adalah 600.

Bidang	Deskripsi	Catatan
`networkProxy`	Opsional. Sebuah objek yang mendefinisikan server proksi untuk terhubung ke.	Server proksi dapat HTTP atau HTTPS. Untuk informasi selengkapnya, lihat Connect pada port 443 atau melalui proksi jaringan.

runtime

Bidang	Deskripsi	Catatan
`cgroup`		
`useSystemd`	Indicates whether your device uses `systemd`.	Valid values are `ya` or `tidak`. Run the `check_ggc_dependencies` script in Modul 1 to see if your device uses `systemd`.

kripto

Objek `crypto` tersebut, ditambahkan di v1.7.0, memperkenalkan properti yang mendukung penyimpanan kunci privat pada hardware security module (HSM) melalui PKCS #11 dan penyimpanan rahasia lokal. Lihat informasi yang lebih lengkap di the section called “Integrasi keamanan perangkat keras” dan Men-deploy rahasia ke core. Konfigurasi untuk penyimpanan kunci privat pada HSMs atau di sistem file yang didukung.

Bidang	Deskripsi	Catatan
`caPath`	Jalur absolut ke CA root AWS IoT ini.	Harus berupa URI file dalam bentuk: `file:///absolute/path/to/file` .

Bidang	Deskripsi	Catatan
		Note Pastikan bahwa titik akhir Anda sesuai dengan jenis sertifikat Anda.
`PKCS11`		
`OpenSSLEngine`	Opsional. Jalur absolut ke file `.so` mesin OpenSSL untuk mengaktifkan mendukung PKCS#11 pada OpenSSL.	Harus berupa jalur ke file pada sistem file. Properti ini diperlukan jika Anda menggunakan agen pembaruan Greengrass OTA dengan keamanan perangkat keras. Untuk informasi selengkapnya, lihat the section called “Konfigurasi pembaruan OTA”.
`P11Provider`	Jalur absolut ke perpustakaan libdl-loadable implementasi PKCS#11.	Harus berupa jalur ke file pada sistem file.
`slotLabel`	Slot label yang digunakan untuk mengidentifikasi modul perangkat keras.	Harus sesuai dengan spesifikasi label PKCS#11.
`slotUserPin`	PIN pengguna yang digunakan untuk mengautentikasi core Greengrass ke modul.	Harus memiliki izin yang memadai untuk melakukan C_Sign dengan kunci privat yang dikonfigurasi.
`principals`		

Bidang	Deskripsi	Catatan
`IoTCertificate`	The certificate and private key that the core uses to make requests to AWS IoT.	
`Sertifikat IoT. privateKeyPath`	Jalur ke kunci privat core.	Untuk penyimpanan sistem file, harus berupa URI file dalam bentuk: `file:///`*`absolute/path/to/file`* . Untuk penyimpanan HSM, harus jalur RFC 7512 PKCS#11 yang menentukan label objek.
`IoTCertificate .certificatePath`	Jalur absolut untuk sertifikat perangkat core.	Harus berupa URI file dalam bentuk: `file:///`*`absolute/path/to/file`* .
`MQTT ServerCer tificate`	Opsional. Kunci privat yang menggunakan core dalam kombinasi dengan sertifikat untuk bertindak sebagai server MQTT atau gateway.	

Bidang	Deskripsi	Catatan
`MQTTServerCertificate . privateKeyPath`	Jalur ke kunci privat server MQTT lokal.	Gunakan nilai ini untuk menentukan kunci privat Anda sendiri untuk server MQTT lokal. Untuk penyimpanan sistem file, harus berupa URI file dalam bentuk: `file:///`*`absolute/path/to/file`* . Untuk penyimpanan HSM, harus jalur RFC 7512 PKCS#11 yang menentukan label objek. Jika properti ini dihilangkan, AWS IoT Greengrass memutar kunci berdasarkan pengaturan rotasi Anda. Jika ditentukan, pelanggan bertanggung jawab untuk memutar kunci.
`SecretsManager`	The private key that secures the data key used for encryption. For more information, see Men-deploy rahasia ke core.	

Bidang	Deskripsi	Catatan
`SecretsManager .privateKeyPath`	Jalur ke kunci privat secrets manager lokal.	Hanya kunci RSA yang didukung. Untuk penyimpanan sistem file, harus berupa URI file dalam bentuk: `file:///`*`absolute/path/to/file`* . Untuk penyimpanan HSM, harus jalur RFC 7512 PKCS#11 yang menentukan label objek. Kunci privat harus dibuat dengan menggunakan mekanisme padding PKCS #1 v1.5 ini.

Properti konfigurasi berikut juga didukung:

Bidang	Deskripsi	Catatan
`mqttMaxConnectionRetryInterval`	Opsional. Interval maksimum (dalam detik) antara koneksi retries MQTT jika koneksi terputus.	Tentukan nilai ini sebagai unsigned integer. Default-nya adalah `60`.
`managedRespawn`	Opsional. Mengindikasikan bahwa agen OTA perlu menjalankan kode kustom sebelum pembaruan.	Nilai yang valid adalah `true` atau `false`. Untuk informasi selengkapnya, lihat Perbarui OTA AWS IoT Greengrass perangkat lunak Core.
`writeDirectory`	Opsional. Direktori tulis di mana AWS IoT Greengras	Untuk informasi selengkapnya, lihat Mengonfigurasi

Bidang	Deskripsi	Catatan
	s membuat semua sumber daya baca/tulis.	direktori tulis untuk AWS IoT Greengrass.

GGC v1.6

```
{
   "coreThing": {
       "caPath": "root-ca-pem",
       "certPath": "cloud-pem-crt",
       "keyPath": "cloud-pem-key",
       "thingArn": "arn:aws:iot:region:account-id:thing/core-thing-name",
       "iotHost": "host-prefix.iot.region.amazonaws.com",
       "ggHost": "greengrass.iot.region.amazonaws.com",
       "keepAlive": 600,
       "mqttMaxConnectionRetryInterval": 60
   },
   "runtime": {
       "cgroup": {
           "useSystemd": "yes|no"
       }
   },
   "managedRespawn": true,
   "writeDirectory": "/write-directory"
}
```

> **Note**
>
> Jika Anda menggunakan opsi AWS IoT Greengrass kreasi Grup Default dari konsol, maka `config.json` file di-deployed ke perangkat core dalam keadaan bekerja yang menentukan konfigurasi default.

File `config.json` mendukung properti berikut:

Bidang	Deskripsi	Catatan
`caPath`	Jalur ke AWS IoT root CA tergantung pada `/greengrass-root /certs` direktori.	Simpan file di bawah `/greengrass-root /certs`.
`certPath`	Jalur ke AWS IoT Greengrass sertifikat core tergantung pada `/greengrass-root /certs` direktori.	Simpan file di bawah `/greengrass-root /certs`.
`keyPath`	Jalur ke AWS IoT Greengrass kunci privat core tergantung pada `/greengrass-root /certs` direktori.	Simpan file di bawah `/greengrass-root /certs`.
`thingArn`	Amazon Resource Name (ARN) dari AWS IoT hal yang mewakili AWS IoT Greengrass perangkat core.	Temukan ARN untuk core Anda di konsol AWS IoT Greengrass tersebut di bawah Cores, atau dengan menjalankan `aws greengrass get-core-definition-version` perintah CLI.
`iotHost`	Titik akhir AWS IoT Anda.	Temukan ini di konsol AWS IoT tersebut di bawah Pengaturan, atau dengan menjalankan `aws iot describe-endpoint` perintah CLI.
`ggHost`	Titik akhir AWS IoT Greengrass Anda.	Nilai ini menggunakan format `greengrass.iot. region.amazonaw`

Bidang	Deskripsi	Catatan
		`s.com` . Gunakan wilayah yang sama dengan `iotHost`.
`keepAlive`	Periode `KeepAlive` MQTT, dalam detik.	Ini adalah nilai opsional. Default-nya adalah `600`.
`mqttMaxConnectionRetryInterval`	Interval maksimum (dalam detik) antara koneksi retries MQTT jika koneksi terputus.	Tentukan nilai ini sebagai unsigned integer. Ini adalah nilai opsional. Default-nya adalah `60`.
`useSystemd`	Menunjukkan apakah perangkat Anda menggunakan `systemd`.	Nilai yang valid adalah `yes` atau `no`. Jalankan `check_ggc_dependencies` skrip di Modul 1 untuk melihat apakah perangkat Anda menggunakan `systemd`.
`managedRespawn`	Fitur pembaruan opsional over-the-air (OTA), ini menunjukkan bahwa agen OTA perlu menjalankan kode khusus sebelum pembaruan.	Nilai yang valid adalah `true` atau `false`. Untuk informasi selengkapnya, lihat Perbarui OTA AWS IoT Greengrass perangkat lunak Core.
`writeDirectory`	Direktori tulis di mana AWS IoT Greengrass membuat semua sumber daya baca/tulis.	Ini adalah nilai opsional. Untuk informasi selengkapnya, lihat Mengonfigurasi direktori tulis untuk AWS IoT Greengrass.

GGC v1.5

```
{
```

```
    "coreThing": {
        "caPath": "root-ca-pem",
        "certPath": "cloud-pem-crt",
        "keyPath": "cloud-pem-key",
        "thingArn": "arn:aws:iot:region:account-id:thing/core-thing-name",
        "iotHost": "host-prefix.iot.region.amazonaws.com",
        "ggHost": "greengrass.iot.region.amazonaws.com",
        "keepAlive": 600
    },
    "runtime": {
        "cgroup": {
            "useSystemd": "yes|no"
        }
    },
    "managedRespawn": true
}
```

File `config.json` eksis di */greengrass-root*`/config` dan berisi parameter berikut:

Bidang	Deskripsi	Catatan
`caPath`	Jalur ke AWS IoT root CA tergantung pada */greengrass-root* `/certs` folder.	Simpan file di bawah */greengrass-root* `/certs` folder.
`certPath`	Jalur ke AWS IoT Greengrass sertifikat core tergantung pada */greengrass-root* `/certs` folder.	Simpan file di bawah */greengrass-root* `/certs` folder.
`keyPath`	Jalur ke AWS IoT Greengrass kunci privat core tergantung pada */greengrass-root* `/certs` folder.	Simpan file di bawah */greengrass-root* `/certs` folder.
`thingArn`	Amazon Resource Name (ARN) dari AWS IoT hal	Temukan ARN untuk core Anda di konsol AWS

Bidang	Deskripsi	Catatan
	yang mewakili AWS IoT Greengrass perangkat core.	IoT Greengrass tersebut di bawah Cores, atau dengan menjalankan `aws greengrass get-core-definition-version` perintah CLI.
`iotHost`	Titik akhir AWS IoT Anda.	Temukan ini di AWS IoT konsol di bawah Pengaturan, atau dengan menjalankan `aws iot describe-endpoint` Perintah.
`ggHost`	Titik akhir AWS IoT Greengrass Anda.	Nilai ini menggunakan format `greengrass.iot.` *`region`*`.amazonaws.com` . Gunakan wilayah yang sama dengan `iotHost`.
`keepAlive`	Periode `KeepAlive` MQTT, dalam detik.	Ini adalah nilai opsional. Nilai default-nya adalah 600 detik.
`useSystemd`	Menunjukkan apakah perangkat Anda menggunakan `systemd`.	Nilai yang valid adalah `yes` atau `no`. Jalankan `check_ggc_dependencies` skrip di Modul 1 untuk melihat apakah perangkat Anda menggunakan `systemd`.

Bidang	Deskripsi	Catatan
`managedRespawn`	Fitur pembaruan opsional over-the-air (OTA), ini menunjukkan bahwa agen OTA perlu menjalankan kode khusus sebelum pembaruan.	Untuk informasi selengkap nya, lihat Perbarui OTA AWS IoT Greengrass perangkat lunak Core.

GGC v1.3

```
{
   "coreThing": {
       "caPath": "root-ca-pem",
       "certPath": "cloud-pem-crt",
       "keyPath": "cloud-pem-key",
       "thingArn": "arn:aws:iot:region:account-id:thing/core-thing-name",
       "iotHost": "host-prefix.iot.region.amazonaws.com",
       "ggHost": "greengrass.iot.region.amazonaws.com",
       "keepAlive": 600
   },
   "runtime": {
       "cgroup": {
           "useSystemd": "yes|no"
       }
   },
   "managedRespawn": true
}
```

File `config.json` eksis di */greengrass-root*`/config` dan berisi parameter berikut:

Bidang	Deskripsi	Catatan
`caPath`	Jalur ke AWS IoT root CA tergantung pada */greengrass-root* `/certs` folder.	Simpan file di bawah */greengrass-root* `/certs` folder.

Bidang	Deskripsi	Catatan
`certPath`	Jalur ke AWS IoT Greengrass sertifikat core tergantung pada */greengrass-root* / `certs` folder.	Simpan file di bawah */greengrass-root* / `certs` folder.
`keyPath`	Jalur ke AWS IoT Greengrass kunci privat core tergantung pada */greengrass-root* / `certs` folder.	Simpan file di bawah */greengrass-root* / `certs` folder.
`thingArn`	Amazon Resource Name (ARN) dari AWS IoT hal yang mewakili AWS IoT Greengrass core.	Anda dapat menemukan nilai ini pada konsol AWS IoT Greengrass di bawah definisi untuk hing AWS IoT Anda.
`iotHost`	Titik akhir AWS IoT Anda.	Anda dapat menemukan nilai ini pada konsol AWS IoT di bawah Pengaturan.
`ggHost`	Titik akhir AWS IoT Greengrass Anda.	Anda dapat menemukan nilai ini pada konsol AWS IoT tersebut di bawah Pengaturan dengan didahului `greengrass.` ini.
`keepAlive`	Periode `KeepAlive` MQTT, dalam detik.	Ini adalah nilai opsional. Nilai default-nya adalah 600 detik.

Bidang	Deskripsi	Catatan
`useSystemd`	Bendera biner, jika perangkat Anda menggunakan `systemd`.	Nilai adalah `yes` atau `no`. Gunakan skrip dependensi di Modul 1 untuk melihat apakah perangkat Anda menggunakan `systemd`.
`managedRespawn`	Fitur pembaruan opsional over-the-air (OTA), ini menunjukkan bahwa agen OTA perlu menjalankan kode khusus sebelum pembaruan.	Untuk informasi selengkapnya, lihat Perbarui OTA AWS IoT Greengrass perangkat lunak Core.

GGC v1.1

```
{
   "coreThing": {
       "caPath": "root-ca-pem",
       "certPath": "cloud-pem-crt",
       "keyPath": "cloud-pem-key",
       "thingArn": "arn:aws:iot:region:account-id:thing/core-thing-name",
       "iotHost": "host-prefix.iot.region.amazonaws.com",
       "ggHost": "greengrass.iot.region.amazonaws.com",
       "keepAlive": 600
   },
   "runtime": {
       "cgroup": {
           "useSystemd": "yes|no"
       }
   }
}
```

File `config.json` eksis di */greengrass-root*`/config` dan berisi parameter berikut:

Bidang	Deskripsi	Catatan
`caPath`	Jalur ke AWS IoT root CA tergantung pada */greengrass-root* / `certs` folder.	Simpan file di bawah */greengrass-root* / `certs` folder.
`certPath`	Jalur ke AWS IoT Greengrass sertifikat core tergantung pada */greengrass-root* / `certs` folder.	Simpan file di bawah */greengrass-root* / `certs` folder.
`keyPath`	Jalur ke AWS IoT Greengrass kunci privat core tergantung pada */greengrass-root* / `certs` folder.	Simpan file di bawah */greengrass-root* / `certs` folder.
`thingArn`	Amazon Resource Name (ARN) dari AWS IoT hal yang mewakili AWS IoT Greengrass core.	Anda dapat menemukan nilai ini pada konsol AWS IoT Greengrass di bawah definisi untuk hing AWS IoT Anda.
`iotHost`	Titik akhir AWS IoT Anda.	Anda dapat menemukan nilai ini pada konsol AWS IoT di bawah Pengaturan.
`ggHost`	Titik akhir AWS IoT Greengrass Anda.	Anda dapat menemukan nilai ini pada konsol AWS IoT tersebut di bawah Pengaturan dengan didahului `greengrass.` ini.

Bidang	Deskripsi	Catatan
`keepAlive`	Periode `KeepAlive` MQTT, dalam detik.	Ini adalah nilai opsional. Nilai default-nya adalah 600 detik.
`useSystemd`	Bendera biner, jika perangkat Anda menggunakan `systemd`.	Nilai adalah `yes` atau `no`. Gunakan skrip dependensi di Modul 1 untuk melihat apakah perangkat Anda menggunakan `systemd`.

GGC v1.0

Di AWS IoT Greengrass Core v1.0, `config.json` di-deployed ke *`greengrass-root`*`/configuration`.

```
{
   "coreThing": {
       "caPath": "root-ca-pem",
       "certPath": "cloud-pem-crt",
       "keyPath": "cloud-pem-key",
       "thingArn": "arn:aws:iot:region:account-id:thing/core-thing-name",
       "iotHost": "host-prefix.iot.region.amazonaws.com",
       "ggHost": "greengrass.iot.region.amazonaws.com",
       "keepAlive": 600
   },
   "runtime": {
       "cgroup": {
           "useSystemd": "yes|no"
       }
   }
}
```

File `config.json` eksis di */`greengrass-root`*`/configuration` dan berisi parameter berikut:

Bidang	Deskripsi	Catatan
`caPath`	Jalur ke AWS IoT root CA tergantung pada `/greengrass-root /configuration/certs` folder.	Simpan file di bawah `/greengrass-root /configuration/certs` folder.
`certPath`	Jalur ke AWS IoT Greengrass sertifikat core tergantung pada `/greengrass-root /configuration/certs` folder.	Simpan file di bawah `/greengrass-root /configuration/certs` folder.
`keyPath`	Jalur ke AWS IoT Greengrass kunci privat core tergantung pada `/greengrass-root /configuration/certs` folder.	Simpan file di bawah `/greengrass-root /configuration/certs` folder.
`thingArn`	Amazon Resource Name (ARN) dari AWS IoT hal yang mewakili AWS IoT Greengrass core.	Anda dapat menemukan nilai ini pada konsol AWS IoT Greengrass di bawah definisi untuk hing AWS IoT Anda.
`iotHost`	Titik akhir AWS IoT Anda.	Anda dapat menemukan nilai ini pada konsol AWS IoT di bawah Pengaturan.
`ggHost`	Titik akhir AWS IoT Greengrass Anda.	Anda dapat menemukan nilai ini pada konsol AWS IoT tersebut di bawah Pengaturan dengan

Bidang	Deskripsi	Catatan
		didahului `greengrass.` ini.
`keepAlive`	Periode `KeepAlive` MQTT, dalam detik.	Ini adalah nilai opsional. Nilai default-nya adalah 600 detik.
`useSystemd`	Bendera biner jika perangkat Anda menggunakan `systemd`.	Nilai adalah `yes` atau `no`. Gunakan skrip dependensi di Modul 1 untuk melihat apakah perangkat Anda menggunakan `systemd`.

Titik akhir layanan harus sesuai dengan jenis sertifikat CA root

Titik akhir AWS IoT Core dan AWS IoT Greengrass Anda harus sesuai dengan jenis sertifikat dari sertifikat CA root pada perangkat Anda. Jika jenis titik akhir dan sertifikat tidak sesuai, upaya autentikasi antara perangkat dan AWS IoT Core atau AWS IoT Greengrass akan gagal. Untuk informasi lebih lanjut, lihat Autentikasi Server dalam Panduan Developer AWS IoT.

Jika perangkat Anda menggunakan sertifikat CA akar Amazon Trust Services (ATS), yang merupakan metode pilihan, perangkat ini juga harus menggunakan titik akhir ATS untuk manajemen perangkat dan pengoperasian bidang data. Titik akhir ATS mencakup segmen `ats`, seperti yang ditunjukkan dalam sintaks berikut untuk titik akhir AWS IoT Core.

```
prefix-ats.iot.region.amazonaws.com
```

> **Note**
>
> Untuk kompatibilitas mundur, AWS IoT Greengrass saat ini mendukung sertifikat CA VeriSign root lama dan titik akhir di beberapa s. Wilayah AWS Jika Anda menggunakan sertifikat CA VeriSign root lama, sebaiknya Anda membuat endpoint ATS dan menggunakan sertifikat CA root ATS sebagai gantinya. Jika tidak, pastikan untuk menggunakan titik akhir warisan yang sesuai. Untuk informasi selengkapnya, lihat Titik akhir warisan yang didukung di. Referensi Umum Amazon Web Services

Titik akhir di config.json

Pada perangkat core Greengrass, titik akhir ditentukan di `coreThing` objek di `config.json` file. Properti `iotHost` mewakili titik akhir AWS IoT Core ini. Properti `ggHost` mewakili titik akhir AWS IoT Greengrass ini. Di contoh potongan berikut, properti ini menentukan titik akhir ATS.

```
{
  "coreThing" : {
    ...
    "iotHost" : "abcde1234uwxyz-ats.iot.us-west-2.amazonaws.com",
    "ggHost" : "greengrass-ats.iot.us-west-2.amazonaws.com",
    ...
  },
```

AWS IoT Coretitik akhir

Anda bisa mendapatkan titik akhir AWS IoT Core Anda dengan menjalankan perintah CLI aws iot describe-endpoint dengan yang sesuai parameter `--endpoint-type` ini.

- Untuk mengembalikan titik akhir yang ditandatangani ATS, jalankan:

  ```
  aws iot describe-endpoint --endpoint-type iot:Data-ATS
  ```

- Untuk mengembalikan titik akhir yang VeriSign ditandatangani lama, jalankan:

  ```
  aws iot describe-endpoint --endpoint-type iot:Data
  ```

AWS IoT Greengrasstitik akhir

Titik akhir AWS IoT Greengrass Anda adalah titik akhir `iotHost` dengan prefiks host digantikan oleh greengrass. Sebagai contoh, titik akhir ATS signed adalah `greengrass-ats.iot.`*`region`*`.amazonaws.com`. Ini menggunakan Wilayah yang sama sebagai titik akhir AWS IoT Core Anda.

Connect pada port 443 atau melalui proksi jaringan

Fitur ini tersedia untuk AWS IoT Greengrass Core v1.7 dan yang lebih baru.

core Greengrass berkomunikasi dengan AWS IoT Core menggunakan protokol pesan MQTT dengan autentikasi klien TLS. Dengan konvensi, MQTT atas TLS menggunakan port 8883. Namun, sebagai

upaya keamanan, lingkungan yang terbatas mungkin akan membatasi lalu lintas masuk dan keluar untuk kisaran kecil port TCP. Sebagai contoh, firewall perusahaan mungkin akan membuka port 443 untuk lalu lintas HTTPS, tetapi menutup port lain yang digunakan untuk protokol yang kurang umum, seperti port 8883 untuk lalu lintas MQTT. Lingkungan lain yang membatasi mungkin mengharuskan semua lalu lintas melalui proksi HTTP sebelum terhubung ke internet.

Untuk mengaktifkan komunikasi dalam skenario ini, AWS IoT Greengrass mengizinkan konfigurasi berikut:

- MQTT dengan autentikasi klien TLS melalui port 443. Jika jaringan Anda mengizinkan koneksi ke port 443, Anda dapat mengonfigurasi core untuk menggunakan port 443 untuk lalu lintas MQTT menggantikan port default 8883. Ini bisa menjadi koneksi langsung ke port 443 atau koneksi melalui server proksi jaringan.

  AWS IoT Greengrass menggunakan metode Application Layer Protocol Network (ALPN) TLS ekstensi untuk mengaktifkan konektor ini. Seperti konfigurasi default, MQTT melalui TLS pada port 443 menggunakan autentikasi klien berbasis sertifikat.

  Ketika dikonfigurasi untuk menggunakan koneksi langsung ke port 443, inti mendukung pembaruan over-the-air (OTA) untuk AWS IoT Greengrass perangkat lunak. Dukungan ini memerlukan AWS IoT Greengrass Core v1.9.3 atau yang lebih baru.
- Komunikasi HTTPS melalui port 443. AWS IoT Greengrass mengirimkan lalu lintas HTTPS melalui port 8443 secara default, tetapi Anda dapat mengonfigurasinya untuk menggunakan port 443.
- Koneksi melalui proksi jaringan. Anda dapat mengonfigurasi server proksi jaringan untuk bertindak sebagai perantara untuk menghubungkan ke core Greengrass. Hanya autentikasi dasar dan HTTP dan HTTPS proxy yang didukung.

  Konfigurasi proksi dilewatkan ke fungsi Lambda yang ditetapkan pengguna melalui `http_proxy`, `https_proxy`, dan `no_proxy` variabel lingkungan. Fungsi user-defined Lambda harus menggunakan pengaturan passed-in ini untuk terhubung melalui proksi. Perpustakaan umum yang digunakan oleh fungsi Lambda untuk membuat koneksi (seperti paket `requests` boto3 atau cURL dan python) biasanya menggunakan variabel lingkungan ini secara default. Jika fungsi Lambda juga menentukan variabel lingkungan yang sama, AWS IoT Greengrass tidak menimpa mereka.

  > **Important**
  >
  > core Greengrass yang dikonfigurasi untuk menggunakan proksi jaringan tidak mendukung pembaruan OTA.

Untuk mengonfigurasi MQTT melalui port 443

Fitur ini memerlukan AWS IoT Greengrass Core v1.7 atau yang lebih baru.

Prosedur ini memungkinkan core Greengrass untuk menggunakan port 443 untuk pesan MQTT dengan AWS IoT Core.

1. Jalankan perintah berikut untuk menghentikan Greengrass daemon:

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

2. Buka *greengrass-root*`/config/config.json` untuk diedit sebagai pengguna su.
3. Di `coreThing` objek, tambahkan `iotMqttPort` properti dan atur nilai untuk **`443`**, seperti yang ditunjukkan di contoh berikut.

```
{
    "coreThing" : {
        "caPath" : "root.ca.pem",
        "certPath" : "12345abcde.cert.pem",
        "keyPath" : "12345abcde.private.key",
        "thingArn" : "arn:aws:iot:us-west-2:123456789012:thing/core-thing-name",
        "iotHost" : "abcd123456wxyz-ats.iot.us-west-2.amazonaws.com",
        "iotMqttPort" : 443,
        "ggHost" : "greengrass-ats.iot.us-west-2.amazonaws.com",
        "keepAlive" : 600
    },
    ...
}
```

4. Mulai daemon.

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

Untuk mengonfigurasi HTTPS melalui port 443

Fitur ini memerlukan AWS IoT Greengrass Core v1.8 atau yang lebih baru.

Prosedur ini mengonfigurasi core untuk menggunakan port 443 untuk komunikasi HTTPS.

1. Jalankan perintah berikut untuk menghentikan Greengrass daemon:

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

2. Buka *greengrass-root*`/config/config.json` untuk diedit sebagai pengguna su.
3. Di `coreThing` objek, tambahkan `iotHttpPort` dan `ggHttpPort` properti, seperti yang ditunjukkan di contoh berikut.

```
{
    "coreThing" : {
        "caPath" : "root.ca.pem",
        "certPath" : "12345abcde.cert.pem",
        "keyPath" : "12345abcde.private.key",
        "thingArn" : "arn:aws:iot:us-west-2:123456789012:thing/core-thing-name",
        "iotHost" : "abcd123456wxyz-ats.iot.us-west-2.amazonaws.com",
        "iotHttpPort" : 443,
        "ggHost" : "greengrass-ats.iot.us-west-2.amazonaws.com",
        "ggHttpPort" : 443,
        "keepAlive" : 600
    },
    ...
}
```

4. Mulai daemon.

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

Untuk mengonfigurasi proksi jaringan

Fitur ini memerlukan AWS IoT Greengrass Core v1.7 atau yang lebih baru.

Prosedur ini mengizinkan AWS IoT Greengrass untuk terhubung ke internet melalui proksi jaringan HTTP atau HTTPS.

1. Jalankan perintah berikut untuk menghentikan Greengrass daemon:

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

2. Buka *greengrass-root*/config/config.json untuk diedit sebagai pengguna su.
3. Di coreThing objek, tambahkan objek networkProxy, seperti yang ditunjukkan dalam contoh berikut.

```
{
    "coreThing" : {
        "caPath" : "root.ca.pem",
        "certPath" : "12345abcde.cert.pem",
        "keyPath" : "12345abcde.private.key",
        "thingArn" : "arn:aws:iot:us-west-2:123456789012:thing/core-thing-name",
        "iotHost" : "abcd123456wxyz-ats.iot.us-west-2.amazonaws.com",
        "ggHost" : "greengrass-ats.iot.us-west-2.amazonaws.com",
        "keepAlive" : 600,
        "networkProxy": {
            "noProxyAddresses" : "http://128.12.34.56,www.mywebsite.com",
            "proxy" : {
                "url" : "https://my-proxy-server:1100",
                "username" : "Mary_Major",
                "password" : "pass@word1357"
            }
        }
    },
    ...
}
```

4. Mulai daemon.

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

objek networkProxy

Gunakan networkProxy objek untuk menentukan informasi tentang proksi jaringan. Objek ini memiliki sifat berikut.

Bidang	Deskripsi
`noProxyAddresses`	Opsional. Daftar comma-separated alamat IP atau nama host yang dibebaskan dari proksi.

`proxy`	Proksi untuk terhubung ke. Proksi memiliki sifat berikut.
	• `url`. URL server proksi, dalam format `scheme://userinfo@host:port` .
	  • `scheme`. Skema. Harus `http` atau `https`.
	  • `userinfo`. Opsional. Nama pengguna dan informasi kata sandi. Jika ditentuka n, `username` dan `password` bidang diabaikan.
	  • `host`. Nama host atau alamat IP server proksi.
	  • `port`. Opsional. Nomor port. Jika tidak ditentukan, nilai default berikut digunakan:
	  • `http`: 80
	  • `https`: 443
	• `username`. Opsional. Nama pengguna yang digunakan untuk mengautentikasi ke server proksi.
	• `password`. Opsional. Password yang digunakan untuk mengautentikasi ke server proksi.

Mengizinkan titik akhir

Komunikasi antara perangkat Greengrass dan AWS IoT Core atau AWS IoT Greengrass harus diautentikasi. Autentikasi ini didasarkan pada sertifikat perangkat X.509 terdaftar dan kunci kriptografi. Untuk mengizinkan permintaan diautentifikasi untuk melewati proksi tanpa enkripsi tambahan, mengizinkan titik akhir berikut.

Titik Akhir	Port	Deskripsi
`greengrass. region.amazonaws.com`	443	Digunakan untuk operasi pesawat kontrol untuk manajemen grup.

`prefix-ats.iot. region.amazonaws.com`
 	
atau
 	
`prefix.iot.region.amazonaws.com`
	MQTT: 8883 atau 443
	
	HTTPS: 8443 atau 443
		Digunakan untuk operasi pesawat data untuk manajemen perangkat, seperti sinkronisasi bayangan.
		
		Izinkan penggunaan salah satu atau kedua titik akhir, tergantung pada apakah perangkat inti dan klien Anda menggunakan sertifikat root CA Amazon Trust Services (pilihan), sertifika

Titik Akhir	Port	Deskripsi
		t CA root lama, atau keduanya. Untuk informasi selengkapnya, lihat the section called “Titik akhir Anda harus sesuai dengan jenis sertifikat”.

Titik Akhir	Port	Deskripsi

`greengrass-ats.iot. region.amazonaws.com`
 	
atau
 	
`greengrass.iot. region.amazonaws.com`
	8443 atau 443	Digunakan untuk operasi penemuan perangkat.
		
		Izinkan penggunaan salah satu atau kedua titik akhir, tergantung pada apakah perangkat inti dan klien Anda menggunakan sertifikat root CA Amazon Trust Services (pilihan), sertifikat CA root lama, atau keduanya. Untuk informasi selengkapnya, lihat the section called “Titik akhir Anda harus sesuai

Titik Akhir	Port	Deskripsi
		dengan jenis sertifikat”. **Note** Klien yang terhubung pada port 443 harus mengimplementasikan Application Layer Protocol Negotiation (ALPN) TLS ekstensi dan lulus `x-amzn-http-ca` sebagai `ProtocolName` di

Titik Akhir	Port	Deskripsi
		`ProtocolNameList`. Untuk informasi selengkapnya, lihat Protokol di AWS IoT Panduan Pengembang.
`*.s3.amazonaws.com`	443	Digunakan untuk operasi penyebaran dan over-the-air pembaruan. Format ini mencakup karakter * karena prefiks titik akhir dikendalikan secara internal dan mungkin berubah setiap saat.

Titik Akhir	Port	Deskripsi
`logs.region.amazonaws.com`	443	Diperlukan jika grup Greengrass dikonfigurasi untuk menulis log. CloudWatch

Mengonfigurasi direktori tulis untuk AWS IoT Greengrass

Fitur ini tersedia untuk AWS IoT Greengrass Core v1.6 dan yang lebih baru.

Secara default, AWS IoT Greengrass perangkat lunak Core diterapkan di bawah direktori root tunggal di mana AWS IoT Greengrass melakukan semua operasi membaca dan menulis. Namun, Anda dapat mengkonfigurasi AWS IoT Greengrass untuk menggunakan direktori terpisah untuk semua operasi tulis, termasuk membuat direktori dan file. Dalam kasus ini, AWS IoT Greengrass menggunakan dua direktori tingkat atas:

- Direktori *greengrass-root* ini, yang dapat Anda tinggalkan sebagai baca-tulis atau opsional membuat read-only. Ini berisi AWS IoT Greengrass perangkat lunak Core dan komponen penting lainnya yang harus tetap tidak berubah selama runtime, seperti sertifikat dan `config.json`.
- Direktori tulis yang ditentukan. Ini berisi konten yang dapat ditulis, seperti log, informasi keadaan, dan di-deployed fungsi Lambda yang ditetapkan pengguna.

Konfigurasi ini menghasilkan struktur direktori berikut.

Direktori root Greengrass

```
greengrass-root/
|-- certs/
|   |-- root.ca.pem
|   |-- hash.cert.pem
|   |-- hash.private.key
|   |-- hash.public.key
|-- config/
|   |-- config.json
```

```
|-- ggc/
|   |-- packages/
|       |-- package-version/
|           |-- bin/
|               |-- daemon
|           |-- greengrassd
|           |-- lambda/
|           |-- LICENSE/
|           |-- release_notes_package-version.html
|               |-- runtime/
|                   |-- java8/
|                   |-- nodejs8.10/
|                   |-- python3.8/
|   |-- core/
```

Tulis Direktori

```
write-directory/
|-- packages/
|   |-- package-version/
|       |-- ggc_root/
|       |-- rootfs_nosys/
|       |-- rootfs_sys/
|       |-- var/
|-- deployment/
|   |-- group/
|       |-- group.json
|   |-- lambda/
|   |-- mlmodel/
|-- var/
|   |-- log/
|   |-- state/
```

Untuk mengonfigurasi direktori tulis

1. Jalankan perintah berikut untuk menghentikan AWS IoT Greengrass daemon:

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

2. Buka *greengrass-root*/config/config.json untuk diedit sebagai pengguna su.
3. Tambahkan `writeDirectory` sebagai parameter dan tentukan jalur ke direktori target, seperti yang ditunjukkan di contoh berikut.

```
{
    "coreThing": {
        "caPath": "root-CA.pem",
        "certPath": "hash.pem.crt",
        ...
    },
    ...
    "writeDirectory" : "/write-directory"
}
```

> **Note**
>
> Anda dapat memperbarui `writeDirectory` pengaturan sesering yang Anda inginkan. Setelah pengaturan diperbarui, AWS IoT Greengrass menggunakan direktori tulis yang baru ditentukan pada permulaan berikutnya, tetapi tidak memigrasi konten dari direktori tulis sebelumnya.

4. Sekarang bahwa direktori tulis Anda dikonfigurasi, Anda dapat secara opsional membuat *greengrass-root* directory read-only. Untuk instruksi, lihat. Untuk Membuat Direktori Root Greengrass Hanya Baca.

   Jika tidak, mulai AWS IoT Greengrass daemon:

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

Untuk membuat direktori root Greengrass hanya baca

Ikuti langkah-langkah ini hanya jika Anda ingin membuat direktori root Greengrass hanya baca. Direktori tulis harus dikonfigurasi sebelum Anda mulai.

1. Berikan izin akses ke direktori yang diperlukan:

a. Memberikan izin baca dan tulis ke `config.json` pemilik.

```
sudo chmod 0600 /greengrass-root/config/config.json
```

b. Membuat ggc_user pemilik sertifikat dan direktori sistem Lambda.

```
sudo chown -R ggc_user:ggc_group /greengrass-root/certs/
sudo chown -R ggc_user:ggc_group /greengrass-root/ggc/packages/1.11.6/lambda/
```

> **Note**
>
> Akun ggc_user dan ggc_group digunakan secara default untuk menjalankan fungsi sistem Lambda. Jika Anda mengonfigurasi tingkat grup identitas akses default untuk menggunakan akun yang berbeda, Anda harus memberikan izin kepada pengguna tersebut (UID) dan grup (GID) sebagai gantinya.

2. Membuat *direktori greengrass-root* read-only dengan menggunakan mekanisme pilihan Anda.

> **Note**
>
> Salah satu cara untuk membuat direktori hanya-baca *greengrass-root* adalah untuk memasang direktori sebagai hanya-baca. Namun, untuk menerapkan pembaruan over-the-air (OTA) ke perangkat lunak AWS IoT Greengrass Core di direktori yang dipasang, direktori harus dilepas terlebih dahulu, dan kemudian dipasang kembali setelah pembaruan. Anda dapat menambahkan `umount` dan `mount` operasi `ota_pre_update` dan `ota_post_update` skrip. Untuk informasi lebih lanjut tentang pembaruan OTA, lihat the section called “Agen pembaruan OTA Greengrass” dan the section called “Respawn terkelola dengan pembaruan OTA”.

3. Mulai daemon.

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

Jika izin dari langkah 1 tidak diatur dengan benar, daemon tidak akan dimulai.

Konfigurasikan pengaturan MQTT

Di AWS IoT Greengrass lingkungan, perangkat klien lokal, fungsi Lambda, konektor, dan komponen sistem dapat berkomunikasi satu sama lain dan dengan. AWS IoT Core Semua komunikasi berjalan melalui core, yang mengelola langganan yang mengotorisasi komunikasi MQTT antar entitas.

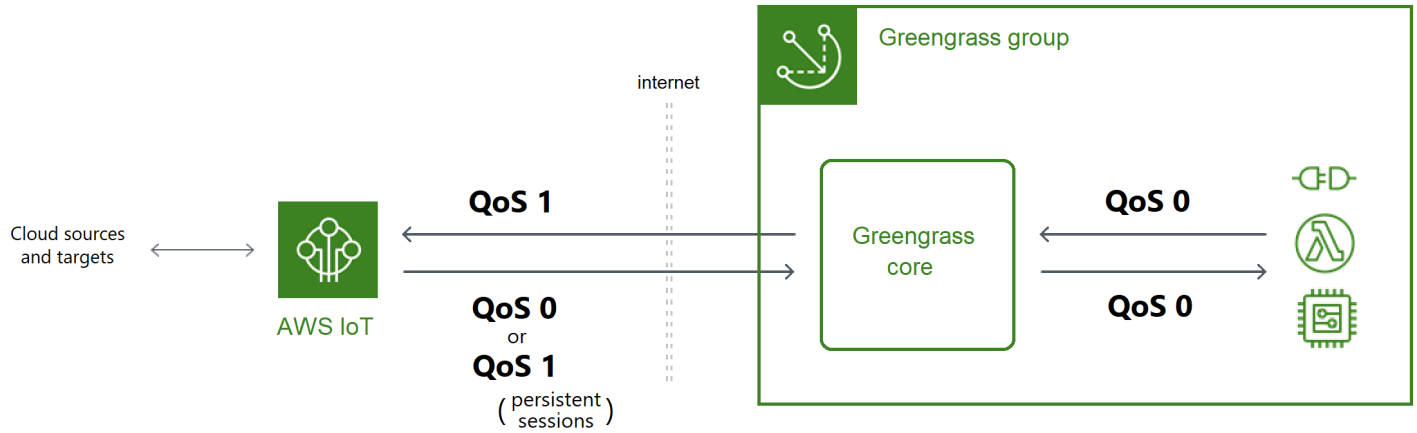
Untuk informasi tentang pengaturan MQTT Anda dapat mengonfigurasi untuk AWS IoT Greengrass, lihat bagian berikut:

- the section called “Kualitas layanan pesan”
- the section called “Antrean pesan MQTT”
- the section called “Sesi persisten MQTT dengan AWS IoT Core”
- the section called “ID klien untuk koneksi MQTT dengan AWS IoT”
- Port MQTT untuk olahpesan lokal
- the section called “Waktu habis untuk menerbitkankan, berlangganan, berhenti berlangganan operasi di koneksi MQTT dengan AWS Cloud”

> **Note**
>
> OPC-UA adalah standar pertukaran informasi untuk komunikasi industri. Untuk mengimplementasikan dukungan OPC-UA pada inti Greengrass, Anda dapat menggunakan konektor IoT. SiteWise Konektor mengirimkan data perangkat industri dari server OPC-UA ke properti aset di AWS IoT SiteWise.

Kualitas layanan pesan

AWS IoT Greengrass mendukung kualitas layanan (QoS) tingkat 0 atau 1, tergantung pada konfigurasi Anda dan target dan arah komunikasi. core Greengrass bertindak sebagai klien untuk komunikasi dengan AWS IoT Core dan broker pesan untuk komunikasi di jaringan lokal.



Untuk informasi lebih lanjut tentang MQTT dan QoS, lihat Memulai di situs web MQTT.

Komunikasi dengan AWS Cloud

- Pesan keluar menggunakan QoS 1

  Core mengirimkan pesan ditujukan untuk AWS Cloud target menggunakan QoS 1. AWS IoT Greengrass menggunakan antrean pesan MQTT untuk memproses pesan ini. Jika pengiriman pesan tidak dikonfirmasi oleh AWS IoT, pesan akan spooled untuk dicoba lagi nanti. Pesan tidak dapat dicoba lagi jika antrean penuh. Konfirmasi pengiriman pesan dapat membantu meminimalkan kehilangan data dari konektivitas intermiten.

  Karena pesan keluar ke AWS IoT menggunakan QoS 1, tingkat maksimum di mana core Greengrass dapat mengirim pesan tergantung pada latensi antara core dan AWS IoT. Setiap kali core mengirimkan pesan, maka menunggu sampai AWS IoT mengakui pesan sebelum mengirim pesan berikutnya. Misalnya, jika waktu pulang-pergi antara core dan waktunya Wilayah AWS adalah 50 milidetik, core dapat mengirim hingga 20 pesan per detik. Pertimbangkan perilaku ini ketika Anda memilih Wilayah AWS di mana core Anda terhubung. Untuk menyerap data IoT volume tinggi ke AWS Cloud, Anda dapat menggunakan Pengelola aliran.

  Untuk informasi selengkapnya tentang antrean pesan MQTT, termasuk cara mengonfigurasi cache penyimpanan lokal yang dapat bertahan pesan yang ditujukan untuk AWS Cloud target, lihat the section called “Antrean pesan MQTT”.

- Pesan masuk menggunakan QoS 0 (default) atau QoS 1

  Secara default, core berlangganan dengan QoS 0 untuk pesan dari AWS Cloud sumber. Jika Anda mengaktifkan sesi persisten, core berlangganan dengan QoS 1. Hal ini dapat membantu

meminimalkan kehilangan data dari konektivitas intermiten. Untuk mengelola QoS untuk langganan ini, Anda mengkonfigurasi pengaturan persistensi pada komponen sistem spooler lokal.

Untuk informasi lebih lanjut, termasuk cara mengaktifkan core untuk membuat sesi persisten dengan AWS Cloud target, lihat the section called “Sesi persisten MQTT dengan AWS IoT Core”.

Komunikasi dengan target lokal

Semua komunikasi lokal menggunakan QoS 0. Inti membuat satu upaya untuk mengirim pesan ke target lokal, yang dapat berupa fungsi Greengrass Lambda, konektor, atau perangkat klien. Inti tidak menyimpan pesan atau mengonfirmasi pengiriman. Pesan dapat diputus di mana saja antara komponen.

> **Note**
>
> Meskipun komunikasi langsung antara fungsi Lambda tidak menggunakan pesan MQTT, perilaku adalah sama.

Antrean pesan MQTT untuk target cloud

MQTT pesan yang ditujukan untuk AWS Cloud target antri untuk menunggu pemrosesan. Pesan antrean diproses di urutan pertama dalam, pertama keluar (FIFO). Setelah pesan diproses dan dipublikasikan ke AWS IoT Core, pesan akan dihapus dari antrean.

Secara default, inti Greengrass menyimpan dalam memori pesan yang belum diproses yang ditujukan untuk target. AWS Cloud Anda dapat mengkonfigurasi core untuk menyimpan pesan yang belum diproses dalam cache penyimpanan lokal sebagai gantinya. Tidak seperti penyimpanan dalam memori, cache penyimpanan lokal memiliki kemampuan untuk bertahan di restart core (sebagai contoh, setelah grup deployment atau reboot perangkat), sehingga AWS IoT Greengrass dapat terus memproses pesan. Anda juga dapat mengonfigurasi ukuran penyimpanan.

> **Warning**
>
> Inti Greengrass mungkin mengantri duplikat pesan MQTT ketika kehilangan koneksi, karena ia mencoba ulang operasi publikasi sebelum klien MQTT mendeteksi bahwa itu offline. Untuk menghindari duplikat pesan MQTT untuk target cloud, konfigurasikan `keepAlive`

nilai inti menjadi kurang dari setengah nilainya. `mqttOperationTimeout` Untuk informasi selengkapnya, lihat AWS IoT Greengrass file konfigurasi core.

AWS IoT Greengrass menggunakan komponen sistem spooler (pada `GGCloudSpooler` fungsi Lambda) untuk mengelola antrean pesan. Anda dapat menggunakan yang berikut ini `GGCloudSpooler` variabel lingkungan untuk mengonfigurasi pengaturan penyimpanan.

- GG_CONFIG_STORAGE_TYPE. Lokasi antrean pesan. Berikut adalah nilai yang valid:
  - `FileSystem`. Menyimpan pesan yang belum diproses dalam cache penyimpanan lokal pada disk perangkat core fisik. Ketika core dimulai ulang, pesan antrean dipertahankan untuk diproses. Pesan akan dihapus setelah pesan diproses.
  - `Memory` (default). Menyimpan pesan yang belum diproses dalam memori. Ketika restart core, pesan antrean hilang.

    Opsi ini dioptimalkan untuk perangkat dengan kemampuan perangkat keras terbatas. Saat menggunakan konfigurasi ini, kami sarankan Anda men-deploy grup atau me-restart perangkat ketika gangguan layanan adalah yang terendah.

- GG_CONFIG_MAX_SIZE_BYTES. Ukuran penyimpanannya, dalam byte. Nilai ini dapat berupa integer non-negatif lebih besar dari atau sama dengan 262144 (256 KB); ukuran yang lebih kecil mencegah AWS IoT Greengrass perangkat lunak Core dari awal. Ukuran default-nya adalah 2,5 MB. Bila batas ukuran tercapai, pesan antrean tertua digantikan oleh pesan baru.

> **Note**
>
> Fitur ini tersedia untuk AWS IoT Greengrass Core v1.6 dan yang lebih baru. Versi sebelumnya menggunakan penyimpanan dalam memori dengan ukuran antrean 2,5 MB. Anda tidak dapat mengonfigurasi pengaturan penyimpanan untuk versi sebelumnya.

Untuk cache pesan di penyimpanan lokal

Anda dapat mengonfigurasi AWS IoT Greengrass untuk cache pesan ke sistem file sehingga mereka bertahan di restart core. Untuk melakukannya, Anda men-deploy versi definisi fungsi di mana `GGCloudSpooler` mengatur fungsi menetapkan jenis penyimpanan untuk `FileSystem`. Anda harus

menggunakan AWS IoT Greengrass API untuk mengonfigurasi cache penyimpanan lokal. Anda tidak dapat melakukan ini di konsol.

Prosedur berikut menggunakan perintah `create-function-definition-version`CLI untuk mengkonfigurasi spooler untuk menyimpan pesan antrian ke sistem file. Ini juga mengonfigurasi ukuran antrean 2,6 MB.

1. Dapatkan ID dari grup Greengrass target dan versi grup. Prosedur ini mengasumsikan bahwa ini adalah versi grup dan grup terbaru. Query berikut mengembalikan grup yang paling baru dibuat.

   ```
   aws greengrass list-groups --query "reverse(sort_by(Groups, &CreationTimestamp))
   [0]"
   ```

   Atau, Anda dapat melakukan query berdasarkan nama. Nama grup tidak perlu unik, sehingga beberapa grup mungkin ditampilkan.

   ```
   aws greengrass list-groups --query "Groups[?Name=='MyGroup']"
   ```

   > **Note**
   >
   > Anda juga dapat menemukan nilai-nilai ini di konsol AWS IoT tersebut. ID grup ditampilkan pada halaman Pengaturan grup. ID versi grup ditampilkan di tab Deployment grup.

2. Salin nilai `Id` dan `LatestVersion` dari grup target di dalam output.
3. Dapatkan versi grup terbaru.

   - Ganti *`id-grup`* dengan `Id` yang Anda salin.
   - Ganti *`latest-group-version-id`*dengan `LatestVersion` yang Anda salin.

   ```
   aws greengrass get-group-version \
   --group-id group-id \
   --group-version-id latest-group-version-id
   ```

4. Dari objek `Definition` dalam output, salin `CoreDefinitionVersionArn` dan ARN dari semua komponen grup lainnya kecuali `FunctionDefinitionVersionArn`. Anda menggunakan nilai-nilai ini ketika membuat versi grup baru.

5. Dari `FunctionDefinitionVersionArn` di output, salin ID dari definisi fungsi. ID adalah GUID yang mengikuti `functions` segmen di dalam ARN, seperti yang ditunjukkan dalam contoh berikut.

```
arn:aws:greengrass:us-west-2:123456789012:/greengrass/
definition/functions/bcfc6b49-beb0-4396-b703-6dEXAMPLEcu5/
versions/0f7337b4-922b-45c5-856f-1aEXAMPLEsf6
```

> **Note**
>
> Atau, Anda dapat membuat definisi fungsi dengan menjalankan `create-function-definition` perintah, dan kemudian menyalin ID dari output.

6. Menambahkan versi definisi fungsi untuk definisi fungsi.

   - Ganti *`function-definition-id`*dengan `Id` yang Anda salin untuk definisi fungsi.
   - Ganti *`arbitrary-function-id`*dengan nama untuk fungsi tersebut, seperti**`spooler-function`**.
   - Tambahkan fungsi Lambda yang ingin Anda sertakan dalam versi ini ke `functions` array. Anda dapat menggunakan `get-function-definition-version` perintah untuk mendapatkan fungsi Greengrass Lambda dari versi definisi fungsi yang ada.

> **Warning**
>
> Pastikan Anda menentukan nilai untuk `GG_CONFIG_MAX_SIZE_BYTES` itu lebih besar dari atau sama dengan 262144. Ukuran yang lebih kecil mencegah AWS IoT Greengrass Perangkat lunak Core dari awal.

```
aws greengrass create-function-definition-version \
--function-definition-id function-definition-id \
--functions '[{"FunctionArn":
 "arn:aws:lambda:::function:GGCloudSpooler:1","FunctionConfiguration":
 {"Environment": {"Variables":
{"GG_CONFIG_MAX_SIZE_BYTES":"2621440","GG_CONFIG_STORAGE_TYPE":"FileSystem"}},"Executable":
 "spooler","MemorySize": 32768,"Pinned": true,"Timeout": 3},"Id": "arbitrary-
function-id"}]'
```

> **Note**
>
> Jika Anda sebelumnya mengatur `GG_CONFIG_SUBSCRIPTION_QUALITY` variabel lingkungan untuk mendukung sesi persisten dengan AWS IoT Core, termasuk dalam fungsi instans ini.

7. Salin `Arn` dari versi definisi fungsi dari output.
8. Buat versi grup yang berisi fungsi Lambda sistem.

   - Ganti *`id-grup`* dengan `Id` untuk grup.
   - Ganti *`core-definition-version-arn`*dengan `CoreDefinitionVersionArn` yang Anda salin dari versi grup terbaru.
   - Ganti *`function-definition-version-arn`*dengan `Arn` yang Anda salin untuk versi definisi fungsi baru.
   - Ganti ARN untuk komponen grup lainnya (sebagai contoh, `SubscriptionDefinitionVersionArn` atau `DeviceDefinitionVersionArn`) yang Anda salin dari versi grup terbaru.
   - Hapus parameter yang tidak terpakai. Sebagai contoh, Hapus `--resource-definition-version-arn` jika versi grup Anda tidak berisi sumber daya apa pun.

```
aws greengrass create-group-version \
--group-id group-id \
--core-definition-version-arn core-definition-version-arn \
--function-definition-version-arn function-definition-version-arn \
--device-definition-version-arn device-definition-version-arn \
--logger-definition-version-arn logger-definition-version-arn \
--resource-definition-version-arn resource-definition-version-arn \
--subscription-definition-version-arn subscription-definition-version-arn
```

9. Salin `Version` dari output. Ini adalah ID dari versi grup baru.
10. Men-deploy grup dengan versi grup baru.

    - Ganti *`id-grup`* dengan `Id` yang Anda salin untuk grup.
    - Ganti *`group-version-id`*dengan `Version` yang Anda salin untuk versi grup baru.

```
aws greengrass create-deployment \
--group-id group-id \
--group-version-id group-version-id \
--deployment-type NewDeployment
```

Untuk memperbarui pengaturan penyimpanan, Anda menggunakan AWS IoT Greengrass API untuk membuat versi definisi fungsi baru yang berisi `GGCloudSpooler` fungsi dengan konfigurasi diperbarui. Kemudian tambahkan versi definisi fungsi ke versi grup baru (bersama dengan komponen grup Anda yang lain) dan men-deploy versi grup. Jika Anda ingin memulihkan konfigurasi default, Anda dapat men-deploy versi definisi fungsi yang tidak termasuk `GGCloudSpooler` fungsi.

Fungsi Lambda sistem ini tidak terlihat di konsol. Namun, setelah fungsi ditambahkan ke versi grup terbaru, itu termasuk dalam deployment yang Anda buat dari konsol, kecuali jika Anda menggunakan API untuk mengganti atau menghapusnya.

Sesi persisten MQTT dengan AWS IoT Core

Fitur ini tersedia untuk AWS IoT Greengrass Core v1.10 dan yang lebih baru.

Sebuah core Greengrass dapat membangun sesi persisten dengan AWS IoT broker pesan. Sesi persisten adalah koneksi yang sedang berlangsung yang mengizinkan core untuk menerima pesan yang dikirim saat core offline. Core adalah klien dalam koneksi.

Dalam sesi persisten, AWS IoT pesan broker menyimpan semua langganan core membuat selama koneksi. Jika inti terputus, broker AWS IoT pesan menyimpan pesan yang tidak diakui dan baru diterbitkan sebagai QoS 1 dan ditujukan untuk target lokal, seperti fungsi Lambda dan perangkat klien. Ketika core menghubungkan kembali, sesi persisten dilanjutkan dan AWS IoT pesan broker mengirimkan pesan yang disimpan ke core pada tingkat maksimum 10 pesan per detik. Sesi persisten memiliki periode kedaluwarsa default 1 jam, yang dimulai ketika broker pesan mendeteksi bahwa core terputus. Untuk informasi selengkapnya, lihat sesi persisten MQTT di AWS IoT Panduan Pengembang.

AWS IoT Greengrass menggunakan komponen sistem spooler (pada `GGCloudSpooler` Fungsi Lambda) untuk membuat langganan yang memiliki AWS IoT sebagai sumbernya. Anda dapat menggunakan variabel lingkungan `GGCloudSpooler` berikut untuk mengonfigurasi sesi persisten.

- GG_CONFIG_SUBSCRIPTION_QUALITY. Kualitas langganan yang memiliki AWS IoT sebagai sumbernya. Berikut adalah nilai yang valid:
  - `AtMostOnce` (default). Menonaktifkan sesi persisten. Langganan menggunakan QoS 0.
  - `AtLeastOncePersistent`. Mengaktifkan sesi persisten. Atur `cleanSession` bendera ke `0` di `CONNECT` pesan dan berlangganan dengan QoS 1.

    Pesan yang diterbitkan dengan QoS 1 yang diterima core dijamin dapat mencapai antrean pekerjaan Greengrass daemon's in-memory. Core mengakui pesan setelah ditambahkan ke antrean. Komunikasi berikutnya dari antrean ke target lokal (sebagai contoh, fungsi Greengrass Lambda, konektor, atau perangkat) dikirim sebagai QoS 0. AWS IoT Greengrass tidak menjamin pengiriman ke target lokal.

    > **Note**
    >
    > Anda dapat menggunakan properti konfigurasi maxWorkItemHitung untuk mengontrol ukuran antrian item kerja. Misalnya, Anda dapat meningkatkan ukuran antrean jika beban kerja Anda memerlukan lalu lintas MQTT berat.

    Ketika sesi persisten diaktifkan, core membuka setidaknya satu koneksi tambahan untuk pertukaran pesan MQTT dengan AWS IoT. Untuk informasi selengkapnya, lihat the section called “ID klien untuk koneksi MQTT dengan AWS IoT”.

Untuk mengonfigurasi sesi persisten MQTT

Anda dapat mengonfigurasi AWS IoT Greengrass untuk menggunakan sesi persisten dengan AWS IoT Core. Untuk melakukannya, Anda men-deploy versi definisi fungsi di mana `GGCloudSpooler` fungsi mengatur kualitas berlangganan untuk `AtLeastOncePersistent`. Pengaturan ini berlaku untuk semua langganan yang memiliki AWS IoT Core (`cloud`) sebagai sumber. Anda harus menggunakan AWS IoT Greengrass API untuk mengonfigurasi sesi persisten. Anda tidak dapat melakukan ini di konsol.

Prosedur berikut menggunakan perintah `create-function-definition-version`CLI untuk mengkonfigurasi spooler untuk menggunakan sesi persisten. Dalam prosedur ini, kami menganggap bahwa Anda memperbarui konfigurasi versi grup terbaru dari grup yang ada.

1. Dapatkan ID dari grup Greengrass target dan versi grup. Prosedur ini mengasumsikan bahwa ini adalah versi grup dan grup terbaru. Query berikut mengembalikan grup yang paling baru dibuat.

```
aws greengrass list-groups --query "reverse(sort_by(Groups, &CreationTimestamp))
[0]"
```

Atau, Anda dapat melakukan query berdasarkan nama. Nama grup tidak perlu unik, sehingga beberapa grup mungkin ditampilkan.

```
aws greengrass list-groups --query "Groups[?Name=='MyGroup']"
```

> **Note**
>
> Anda juga dapat menemukan nilai-nilai ini di konsol AWS IoT tersebut. ID grup ditampilkan pada halaman Pengaturan grup. ID versi grup ditampilkan di tab Deployment grup.

2. Salin nilai `Id` dan `LatestVersion` dari grup target di dalam output.
3. Dapatkan versi grup terbaru.

   - Ganti *id-grup* dengan `Id` yang Anda salin.
   - Ganti *latest-group-version-id*dengan `LatestVersion` yang Anda salin.

```
aws greengrass get-group-version \
--group-id group-id \
--group-version-id latest-group-version-id
```

4. Dari objek `Definition` dalam output, salin `CoreDefinitionVersionArn` dan ARN dari semua komponen grup lainnya kecuali `FunctionDefinitionVersionArn`. Anda menggunakan nilai-nilai ini ketika membuat versi grup baru.
5. Dari `FunctionDefinitionVersionArn` di output, salin ID dari definisi fungsi. ID adalah GUID yang mengikuti `functions` segmen di dalam ARN, seperti yang ditunjukkan dalam contoh berikut.

```
arn:aws:greengrass:us-west-2:123456789012:/greengrass/
definition/functions/bcfc6b49-beb0-4396-b703-6dEXAMPLEcu5/
versions/0f7337b4-922b-45c5-856f-1aEXAMPLEsf6
```

> **Note**
>
> Atau, Anda dapat membuat definisi fungsi dengan menjalankan `create-function-definition` perintah, dan kemudian menyalin ID dari output.

6. Menambahkan versi definisi fungsi untuk definisi fungsi.

   - Ganti *`function-definition-id`*dengan `Id` yang Anda salin untuk definisi fungsi.
   - Ganti *`arbitrary-function-id`*dengan nama untuk fungsi tersebut, seperti**`spooler-function`**.
   - Tambahkan fungsi Lambda yang ingin Anda sertakan dalam versi ini ke `functions` array. Anda dapat menggunakan `get-function-definition-version` perintah untuk mendapatkan fungsi Greengrass Lambda dari versi definisi fungsi yang ada.

```
aws greengrass create-function-definition-version \
--function-definition-id function-definition-id \
--functions '[{"FunctionArn":
 "arn:aws:lambda:::function:GGCloudSpooler:1","FunctionConfiguration":
 {"Environment": {"Variables":
{"GG_CONFIG_SUBSCRIPTION_QUALITY":"AtLeastOncePersistent"}},"Executable":
 "spooler","MemorySize": 32768,"Pinned": true,"Timeout": 3},"Id": "arbitrary-
function-id"}]'
```

> **Note**
>
> Jika Anda sebelumnya mengatur `GG_CONFIG_STORAGE_TYPE` atau `GG_CONFIG_MAX_SIZE_BYTES` variabel lingkungan ke menentukan pengaturan penyimpanan, termasuk mereka dalam contoh fungsi ini.

7. Salin `Arn` dari versi definisi fungsi dari output.
8. Buat versi grup yang berisi fungsi Lambda sistem.

   - Ganti *`id-grup`* dengan `Id` untuk grup.
   - Ganti *`core-definition-version-arn`*dengan `CoreDefinitionVersionArn` yang Anda salin dari versi grup terbaru.

- Ganti *function-definition-version-arn*dengan `Arn` yang Anda salin untuk versi definisi fungsi baru.
- Ganti ARN untuk komponen grup lainnya (sebagai contoh, `SubscriptionDefinitionVersionArn` atau `DeviceDefinitionVersionArn`) yang Anda salin dari versi grup terbaru.
- Hapus parameter yang tidak terpakai. Sebagai contoh, Hapus `--resource-definition-version-arn` jika versi grup Anda tidak berisi sumber daya apa pun.

```
aws greengrass create-group-version \
--group-id group-id \
--core-definition-version-arn core-definition-version-arn \
--function-definition-version-arn function-definition-version-arn \
--device-definition-version-arn device-definition-version-arn \
--logger-definition-version-arn logger-definition-version-arn \
--resource-definition-version-arn resource-definition-version-arn \
--subscription-definition-version-arn subscription-definition-version-arn
```

9. Salin `Version` dari output. Ini adalah ID dari versi grup baru.
10. Men-deploy grup dengan versi grup baru.

    - Ganti *id-grup* dengan `Id` yang Anda salin untuk grup.
    - Ganti *group-version-id*dengan `Version` yang Anda salin untuk versi grup baru.

```
aws greengrass create-deployment \
--group-id group-id \
--group-version-id group-version-id \
--deployment-type NewDeployment
```

11. (Opsional) Meningkatkan properti maxWorkItemCount dalam file konfigurasi inti. Hal ini dapat membantu menangani core meningkatkan lalu lintas MQTT dan komunikasi dengan target lokal.

Untuk memperbarui core dengan perubahan konfigurasi ini, Anda menggunakan AWS IoT Greengrass API untuk membuat versi definisi fungsi baru yang berisi `GGCloudSpooler` fungsi dengan konfigurasi diperbarui. Kemudian tambahkan versi definisi fungsi ke versi grup baru (bersama dengan komponen grup Anda yang lain) dan men-deploy versi grup. Jika Anda ingin

memulihkan konfigurasi default, Anda dapat membuat versi definisi fungsi yang tidak menyertakan `GGCloudSpooler` fungsi.

Fungsi Lambda sistem ini tidak terlihat di konsol. Namun, setelah fungsi ditambahkan ke versi grup terbaru, itu termasuk dalam deployment yang Anda buat dari konsol, kecuali jika Anda menggunakan API untuk mengganti atau menghapusnya.

ID klien untuk koneksi MQTT dengan AWS IoT

Fitur ini tersedia untuk AWS IoT Greengrass Core v1.8 dan yang lebih baru.

core Greengrass membuka koneksi MQTT dengan AWS IoT Core untuk operasi seperti sinkronisasi bayangan dan manajemen sertifikat. Untuk koneksi ini, core menghasilkan ID klien diprediksi berdasarkan nama core. ID klien yang dapat diprediksi dapat digunakan dengan fitur pemantauan, audit, dan penetapan harga, termasuk peristiwa AWS IoT Device Defender AWS IoTsiklus hidup. Anda juga dapat membuat logika di sekitar ID klien yang dapat diprediksi (sebagai contoh, kebijakan berlangganan templat berdasarkan atribut sertifikat).

GGC v1.9 and later

Dua komponen sistem Greengrass membuka koneksi MQTT dengan AWS IoT Core. Komponen ini menggunakan pola berikut untuk menghasilkan ID klien untuk koneksi.

Operasi	Pola ID klien
Deployment	*core-thing-name* Contoh: `MyCoreThing` Gunakan ID klien ini untuk notifikasi kejadian connect, disconnect, berlangganan, dan berhenti berlangganan.
Langganan	*core-thing-name* `-c`*n* Contoh: `MyCoreThing-c01` *n* adalah integer yang dimulai pada 00 dan bertahap dengan setiap koneksi baru ke jumlah maksimum 250. Jumlah koneksi ditentukan oleh jumlah perangkat yang

Operasi	Pola ID klien
	menyinkronkan keadaan bayangan mereka dengan AWS IoT Core (maksimum 2.500 per grup) dan jumlah langganan dengan `cloud` sebagai sumber mereka di grup (maksimum 10.000 per grup). Komponen sistem spooler terhubung dengan AWS IoT Core untuk bertukar pesan untuk langganan dengan sumber cloud atau target. Spooler juga bertindak sebagai proksi untuk pertukaran pesan antara AWS IoT Core dan layanan bayangan lokal dan certificate manager perangkat.

Untuk menghitung jumlah koneksi MQTT per grup, gunakan rumus berikut:

```
number of MQTT connections per group = number of connections for
Deployment Agent + number of connections for Subscriptions
```

Di mana,

- jumlah koneksi untuk Deployment Agent = 1.
- jumlah koneksi untuk Langganan = `(2 subscriptions for supporting certificate generation + number of MQTT topics in AWS IoT Core + number of device shadows synced) / 50`.
  - Di mana, `50` = jumlah maksimum langganan per koneksi yang AWS IoT Core dapat mendukung.

> **Note**
>
> Jika Anda mengaktifkan sesi persisten untuk langganan dengan AWS IoT Core, koneksi membuka setidaknya satu koneksi tambahan untuk digunakan dalam sesi persisten. Komponen sistem tidak mendukung sesi persisten, sehingga mereka tidak dapat berbagi koneksi tersebut.

Untuk mengurangi jumlah koneksi MQTT dan membantu mengurangi biaya, Anda dapat menggunakan fungsi Lambda lokal untuk mengumpulkan data di edge. Kemudian Anda mengirim data gabungan ke AWS Cloud. Sebagai hasilnya, Anda menggunakan lebih sedikit topik MQTT di AWS IoT Core. Untuk informasi selengkapnya, silakan lihat Harga AWS IoT Greengrass.

GGC v1.8

Beberapa komponen sistem Greengrass membuka koneksi MQTT dengan AWS IoT Core. Komponen ini menggunakan pola berikut untuk menghasilkan ID klien untuk koneksi.

Operasi	Pola ID klien
Deployment	*core-thing-name* Contoh: `MyCoreThing` Gunakan ID klien ini untuk notifikasi kejadian connect, disconnect, berlangganan, dan berhenti berlangganan.
MQTT pertukaran pesan dengan AWS IoT Core	*core-thing-name* `-spr` Contoh: `MyCoreThing-spr`
Sinkronisasi bayangan	*core-thing-name* `-s`*nn* Contoh: `MyCoreThing-s01` *nn* adalah integer yang dimulai pada 00 dan bertahap dengan setiap koneksi baru untuk maksimum 03. Jumlah koneksi ditentukan oleh jumlah perangkat (maksimum 200 perangkat per grup) yang menyinkronkan keadaan bayangan mereka dengan AWS IoT Core (maksimum 50 langganan setiap koneksi).
Manajemen sertifikat perangkat lunak	*core-thing-name* `-dcm` Contoh: `MyCoreThing-dcm`

> **Note**
>
> Duplikat klien ID yang digunakan dalam koneksi simultan dapat menyebabkan loop connect-disconnect tak terbatas. Hal ini dapat terjadi jika perangkat lain hardcoded untuk menggunakan nama perangkat core sebagai ID klien di koneksi. Untuk informasi lebih lanjut, lihat langkah pemecahan masalah.

Perangkat Greengrass juga terintegrasi sepenuhnya dengan layanan Fleet Indexing AWS IoT Device Management. Hal ini memungkinkan Anda untuk indeks dan mencari perangkat berdasarkan atribut perangkat, keadaan bayangan, dan status koneksi di cloud. Sebagai contoh, perangkat Greengrass membangun setidaknya satu koneksi yang menggunakan nama hal sebagai ID klien, sehingga Anda dapat menggunakan pengindeksan konektivitas perangkat untuk menemukan perangkat Greengrass yang saat ini terhubung atau terputus ke AWS IoT Core. Untuk informasi selengkapnya, lihat layanan pengindeksan armada di AWS IoT Panduan Pengembang.

Konfigurasi port MQTT untuk pesan lokal

Fitur ini memerlukan AWS IoT Greengrass Core v1.10 atau yang lebih baru.

Inti Greengrass bertindak sebagai broker pesan lokal untuk pesan MQTT antara fungsi Lambda lokal, konektor, dan perangkat klien. Secara default, core menggunakan port 8883 untuk lalu lintas MQTT pada jaringan lokal. Anda mungkin ingin mengubah port untuk menghindari konflik dengan perangkat lunak lain yang berjalan pada port 8883.

Untuk mengkonfigurasi nomor port yang menggunakan contoh untuk lalu lintas MQTT lokal

1. Jalankan perintah berikut untuk menghentikan Greengrass daemon:

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

2. Buka *greengrass-root*`/config/config.json` untuk diedit sebagai pengguna su.
3. Di `coreThing` objek, tambahkan `ggMqttPort` properti dan tetapkan nilai ke nomor port yang ingin Anda gunakan. Nilai yang valid adalah 1024 sampai 65535. Contoh berikut menetapkan nomor port menjadi `9000`.

   ```
   {
       "coreThing" : {
   ```

```
        "caPath" : "root.ca.pem",
        "certPath" : "12345abcde.cert.pem",
        "keyPath" : "12345abcde.private.key",
        "thingArn" : "arn:aws:iot:us-west-2:123456789012:thing/core-thing-name",
        "iotHost" : "abcd123456wxyz-ats.iot.us-west-2.amazonaws.com",
        "ggHost" : "greengrass-ats.iot.us-west-2.amazonaws.com",
        "ggMqttPort" : 9000,
        "keepAlive" : 600
    },
    ...
}
```

4. Mulai daemon.

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

5. Jika deteksi IP otomatis diaktifkan untuk core, konfigurasi selesai.

   Jika deteksi IP otomatis tidak diaktifkan, Anda harus memperbarui informasi konektivitas untuk core. Hal ini memungkinkan perangkat klien untuk menerima nomor port yang benar selama operasi penemuan untuk memperoleh informasi konektivitas inti. Anda dapat menggunakan AWS IoT konsol atau AWS IoT Greengrass API untuk memperbarui informasi konektivitas core. Untuk prosedur ini, Anda memperbarui hanya nomor port. Alamat IP lokal untuk core tetap sama.

   Untuk memperbarui informasi konektivitas untuk inti (konsol)

   1. Pada halaman konfigurasi grup, pilih inti Greengrass.
   2. Pada halaman detail inti, pilih tab titik akhir broker MQTT.
   3. Pilih Kelola titik akhir dan kemudian pilih Tambah titik akhir
   4. Masukkan alamat IP lokal Anda saat ini dan nomor port baru. Contoh berikut mengatur nomor port `9000` untuk alamat IP `192.168.1.8`.
   5. Hapus titik akhir usang, dan kemudian pilih Perbarui

   Untuk memperbarui informasi konektivitas untuk inti (API)

   - Gunakan UpdateConnectivityInfotindakan. Contoh berikut menggunakan `update-connectivity-info` di AWS CLI untuk mengatur nomor port `9000` untuk alamat IP `192.168.1.8`.

```
aws greengrass update-connectivity-info \
    --thing-name "MyGroup_Core" \
    --connectivity-info "[{\"Metadata\":\"\",\"PortNumber\":9000,
\"HostAddress\":\"192.168.1.8\",\"Id\":\"localIP_192.168.1.8\"},{\"Metadata
\":\"\",\"PortNumber\":8883,\"HostAddress\":\"127.0.0.1\",\"Id\":
\"localhost_127.0.0.1_0\"}]"
```

> **Note**
>
> Anda juga dapat mengkonfigurasi port yang menggunakan core untuk pesan MQTT dengan AWS IoT Core. Untuk informasi selengkapnya, lihat the section called “Connect pada port 443 atau melalui proksi jaringan”.

Waktu habis untuk menerbitkankan, berlangganan, berhenti berlangganan operasi di koneksi MQTT dengan AWS Cloud

Fitur ini tersedia di AWS IoT Greengrass v1.10.2 atau yang lebih baru.

Anda dapat mengkonfigurasi jumlah waktu (dalam detik) untuk mengizinkan core Greengrass untuk menyelesaikan penerbitan, berlangganan, atau berhenti berlangganan operasi di koneksi MQTT ke AWS IoT Core. Anda mungkin ingin menyesuaikan pengaturan ini jika waktu operasi habis karena kendala bandwidth atau latency tinggi. Untuk mengonfigurasi pengaturan ini di config.json file, tambah atau ubah `mqttOperationTimeout` properti `coreThing` objek. Sebagai contoh:

```
{
  "coreThing": {
    "mqttOperationTimeout": 10,
    "caPath": "root-ca.pem",
    "certPath": "hash.cert.pem",
    "keyPath": "hash.private.key",
    ...
  },
  ...
}
```

Waktu habis default adalah 5 detik. Waktu habis minimal adalah 5 detik.

Aktifkan deteksi IP otomatis

Anda dapat mengonfigurasi AWS IoT Greengrass untuk mengaktifkan perangkat klien dalam grup Greengrass untuk secara otomatis menemukan inti Greengrass. Ketika diaktifkan, core mencatat untuk perubahan ke alamat IP-nya. Jika alamat berubah, core menerbitkan daftar alamat yang diperbarui. Alamat ini tersedia untuk perangkat klien yang berada dalam grup Greengrass yang sama dengan intinya.

> **Note**
>
> AWS IoTKebijakan untuk perangkat klien harus memberikan `greengrass:Discover` izin untuk mengizinkan perangkat mengambil informasi konektivitas untuk inti. Untuk informasi selengkapnya tentang kebijakan laporan, lihat the section called “Otorisasi Discovery”.

Untuk mengaktifkan fitur ini dari AWS IoT Greengrass konsol, pilih Deteksi otomatis ketika Anda men-deploy grup Greengrass Anda untuk pertama kalinya. Anda juga dapat mengaktifkan atau menonaktifkan fitur ini pada halaman konfigurasi grup dengan memilih tab fungsi Lambda dan memilih detektor IP. Deteksi IP otomatis diaktifkan jika Secara otomatis mendeteksi dan mengganti titik akhir broker MQTT dipilih.

Untuk mengelola penemuan otomatis dengan AWS IoT Greengrass API, Anda harus mengkonfigurasi `IPDetector` fungsi sistem Lambda. Prosedur berikut menunjukkan cara menggunakan perintah create-function-definition-versionCLI untuk mengkonfigurasi penemuan otomatis inti Greengrass.

1. Dapatkan ID dari grup Greengrass target dan versi grup. Prosedur ini mengasumsikan bahwa ini adalah versi grup dan grup terbaru. Query berikut mengembalikan grup yang paling baru dibuat.

   ```
   aws greengrass list-groups --query "reverse(sort_by(Groups, &CreationTimestamp))
   [0]"
   ```

   Atau, Anda dapat melakukan query berdasarkan nama. Nama grup tidak perlu unik, sehingga beberapa grup mungkin ditampilkan.

   ```
   aws greengrass list-groups --query "Groups[?Name=='MyGroup']"
   ```

> **Note**
>
> Anda juga dapat menemukan nilai-nilai ini di konsol AWS IoT tersebut. ID grup ditampilkan pada halaman Pengaturan grup. ID versi grup ditampilkan di tab Deployment grup.

2. Salin nilai `Id` dan `LatestVersion` dari grup target di dalam output.
3. Dapatkan versi grup terbaru.

   - Ganti *`id-grup`* dengan `Id` yang Anda salin.
   - Ganti *`latest-group-version-id`*dengan `LatestVersion` yang Anda salin.

```
aws greengrass get-group-version \
--group-id group-id \
--group-version-id latest-group-version-id
```

4. Dari objek `Definition` dalam output, salin `CoreDefinitionVersionArn` dan ARN dari semua komponen grup lainnya kecuali `FunctionDefinitionVersionArn`. Anda menggunakan nilai-nilai ini ketika membuat versi grup baru.
5. Dari `FunctionDefinitionVersionArn` di output, salin ID definisi fungsi dan versi definisi fungsi:

```
arn:aws:greengrass:region:account-id:/greengrass/groups/function-definition-id/
versions/function-definition-version-id
```

> **Note**
>
> Anda dapat secara opsional membuat definisi fungsi dengan menjalankan `create-function-definition`perintah, dan kemudian menyalin ID dari output.

6. Gunakan `get-function-definition-version` perintah untuk mendapatkan keadaan definisi saat ini. Gunakan yang *`function-definition-id`*Anda salin untuk definiton fungsi. Sebagai contoh, *`4d941bc7-92a1-4f45-8d64-EXAMPLEf76c3`*.

```
aws greengrass get-function-definition-version
--function-definition-id function-definition-id
```

```
--function-definition-version-id function-definition-version-id
```

Buat catatan konfigurasi fungsi yang tercantum. Anda harus menyertakan ini saat membuat versi definisi fungsi baru untuk mencegah hilangnya pengaturan definisi Anda saat ini.

7. Tambahkan versi definisi fungsi untuk definisi fungsi.

   - Ganti *function-definition-id*dengan `Id` yang Anda salin untuk definisi fungsi. Sebagai contoh, *4d941bc7-92a1-4f45-8d64-EXAMPLEf76c3*.
   - Ganti *arbitrary-function-id*dengan nama untuk fungsi tersebut, seperti**auto-detection-function**.
   - Tambahkan semua fungsi Lambda yang ingin Anda sertakan dalam versi ini ke `functions` Array, seperti yang tercantum dalam langkah sebelumnya.

```
aws greengrass create-function-definition-version \
--function-definition-id function-definition-id \
--functions
 '[{"FunctionArn":"arn:aws:lambda:::function:GGIPDetector:1","Id":"arbitrary-
function-id","FunctionConfiguration":
{"Pinned":true,"MemorySize":32768,"Timeout":3}}]'\
--region us-west-2
```

8. Salin `Arn` dari versi definisi fungsi dari output.
9. Buat versi grup yang berisi fungsi Lambda sistem.

   - Ganti *id-grup* dengan `Id` untuk grup.
   - Ganti *core-definition-version-arn*dengan `CoreDefinitionVersionArn` yang Anda salin dari versi grup terbaru.
   - Ganti *function-definition-version-arn*dengan `Arn` yang Anda salin untuk versi definisi fungsi baru.
   - Ganti ARN untuk komponen grup lainnya (sebagai contoh, `SubscriptionDefinitionVersionArn` atau `DeviceDefinitionVersionArn`) yang Anda salin dari versi grup terbaru.
   - Hapus parameter yang tidak terpakai. Sebagai contoh, Hapus `--resource-definition-version-arn` jika versi grup Anda tidak berisi sumber daya apa pun.

```
aws greengrass create-group-version \
--group-id group-id \
--core-definition-version-arn core-definition-version-arn \
--function-definition-version-arn function-definition-version-arn \
--device-definition-version-arn device-definition-version-arn \
--logger-definition-version-arn logger-definition-version-arn \
--resource-definition-version-arn resource-definition-version-arn \
--subscription-definition-version-arn subscription-definition-version-arn
```

10. Salin `Version` dari output. Ini adalah ID dari versi grup baru.
11. Men-deploy grup dengan versi grup baru.

    - Ganti *`id-grup`* dengan `Id` yang Anda salin untuk grup.
    - Ganti *`group-version-id`*dengan `Version` yang Anda salin untuk versi grup baru.

```
aws greengrass create-deployment \
--group-id group-id \
--group-version-id group-version-id \
--deployment-type NewDeployment
```

Jika Anda ingin secara manual memasukkan alamat IP core Greengrass Anda, Anda dapat menyelesaikan tutorial ini dengan definisi fungsi yang berbeda yang tidak termasuk `IPDetector` fungsi. Ini akan mencegah fungsi deteksi dari menemukan dan secara otomatis memasukkan alamat IP core Greengrass Anda.

Fungsi Lambda sistem ini tidak terlihat di konsol Lambda. Setelah fungsi ditambahkan ke versi grup terbaru, itu termasuk dalam deployment yang Anda buat dari konsol, kecuali jika Anda menggunakan API untuk mengganti atau menghapusnya.

Konfigurasikan sistem init untuk memulai Greengrass daemon

Ini adalah praktik yang baik untuk mengatur sistem init Anda untuk memulai Greengrass daemon saat boot, terutama ketika mengelola armada besar perangkat.

> **Note**
>
> Jika Anda menggunakan `apt` untuk menginstal AWS IoT Greengrass perangkat lunak Core, Anda dapat menggunakan skrip systemd untuk mengaktifkan start on boot. Untuk informasi selengkapnya, lihat the section called “Gunakan skrip systemd untuk mengelola siklus hidup Greengrass daemon”.

Ada berbagai jenis sistem init, seperti initd, systemd, dan SystemV, dan mereka menggunakan parameter konfigurasi yang sama. Contoh berikut adalah file layanan untuk systemd. Parameter `Type` diatur ke `forking` karena greengrassd (yang digunakan untuk memulai Greengrass) forks proses Greengrass daemon, dan `Restart` parameter diatur ke `on-failure` untuk mengarahkan systemd untuk me-restart Greengrass jika Greengrass memasuki keadaan gagal.

> **Note**
>
> Untuk melihat apakah perangkat Anda menggunakan systemd, jalankan `check_ggc_dependencies` seperti yang dijelaskan di Modul 1. Kemudian untuk menggunakan systemd, pastikan bahwa parameter `useSystemd` di `config.json` diatur ke `yes`.

```
[Unit]
Description=Greengrass Daemon

[Service]
Type=forking
PIDFile=/var/run/greengrassd.pid
Restart=on-failure
ExecStart=/greengrass/ggc/core/greengrassd start
ExecReload=/greengrass/ggc/core/greengrassd restart
ExecStop=/greengrass/ggc/core/greengrassd stop

[Install]
WantedBy=multi-user.target
```

Lihat juga

- Apakah AWS IoT Greengrass itu?

- the section called “Platform dan persyaratan yang didukung”
- Memulai dengan AWS IoT Greengrass
- the section called “Ringkasan tentang model objek grup”
- the section called “Integrasi keamanan perangkat keras”

AWS IoT Greengrass Version 1kebijakan pemeliharaan

Gunakan kebijakan AWS IoT Greengrass V1 pemeliharaan ini untuk memahami berbagai tingkat pemeliharaan dan pembaruan untuk AWS IoT Greengrass V1 layanan dan perangkat lunak AWS IoT Greengrass Core v1.x.

Topik

- AWS IoT Greengrassskema pembuatan versi
- Fase siklus hidup untuk versi utama perangkat lunak Core AWS IoT Greengrass
- Kebijakan pemeliharaan untuk perangkat lunak AWS IoT Greengrass Inti
- Jadwal penghentian
- Kebijakan Support untuk AWS Lambda fungsi pada perangkat inti Greengrass
- Support kebijakan untuk AWS IoT Penguji Perangkat untuk AWS IoT Greengrass V1
- Akhir jadwal pemeliharaan

AWS IoT Greengrassskema pembuatan versi

AWS IoT Greengrassmenggunakan versi semantik untuk perangkat lunak Core. AWS IoT Greengrass Versi semantik mengikuti sistem nomor mayor.minor.patch. Peningkatan versi utama untuk perubahan fungsional dan API yang tidak kompatibel ke belakang dengan versi utama sebelumnya. Versi minor meningkat untuk rilis yang menambahkan fungsionalitas baru yang kompatibel ke belakang. Versi patch meningkat untuk patch keamanan atau perbaikan bug. Sejak rilis besar pertamanya, v1.0.0, AWS IoT Greengrass telah merilis 11 versi minor dari perangkat lunak AWS IoT Greengrass Core v1.x, di mana v1.11.6 adalah rilis terbaru. Kami menyarankan Anda memperbarui perangkat lunak AWS IoT Greengrass Core Anda ke versi terbaru yang tersedia untuk memanfaatkan fitur baru, penyempurnaan, dan perbaikan bug.

Pada Desember 2020, AWS IoT Greengrass merilis pembaruan versi utama pertamanya. Pembaruan ini termasuk AWS IoT Greengrass V2 layanan dan versi 2.0.3 dari perangkat lunak AWS IoT Greengrass Core. Untuk aplikasi baru, kami sangat menyarankan Anda menggunakan AWS IoT Greengrass Version 2 dan perangkat lunak AWS IoT Greengrass Core v2.x. Versi 2 menerima fitur baru, mencakup semua fitur utama V1, dan mendukung platform tambahan dan penyebaran berkelanjutan ke armada perangkat besar. Untuk informasi lebih lanjut, lihat Apa yang dimaksud dengan AWS IoT Greengrass V2?.

Fase siklus hidup untuk versi utama perangkat lunak Core AWS IoT Greengrass

Setiap versi utama perangkat lunak AWS IoT Greengrass Core memiliki tiga fase siklus hidup berurutan berikut. Setiap fase siklus hidup menyediakan tingkat pemeliharaan yang berbeda selama periode waktu setelah tanggal rilis awal.

- Fase rilis - AWS IoT Greengrass dapat merilis pembaruan berikut:
  - Pembaruan versi minor yang menyediakan fitur atau penyempurnaan baru pada fitur yang ada
  - Pembaruan versi patch yang menyediakan tambalan keamanan dan perbaikan bug
- Fase pemeliharaan - AWS IoT Greengrass dapat merilis pembaruan versi tambalan yang menyediakan tambalan keamanan dan perbaikan bug. AWS IoT Greengrasstidak akan merilis fitur atau penyempurnaan baru ke fitur yang ada selama fase pemeliharaan.
- Fase umur yang diperpanjang - AWS IoT Greengrass tidak akan merilis pembaruan yang menyediakan fitur, penyempurnaan pada fitur yang ada, patch keamanan, atau perbaikan bug. Namun, AWS Cloud endpoint dan operasi API akan tetap tersedia dan beroperasi sesuai dengan AWS IoT GreengrassService Level Agreement. Perangkat yang menjalankan perangkat lunak AWS IoT Greengrass Core v1.x dapat terus terhubung ke AWS Cloud dan beroperasi.

Setelah fase masa pakai yang diperpanjang berakhir untuk versi utamaAWS IoT Greengrass, AWS Cloud titik akhir dan operasi API akan usang dan tidak lagi tersedia. Perangkat yang menjalankan perangkat lunak AWS IoT Greengrass Core v1.x tidak akan dapat terhubung ke AWS Cloud layanan untuk beroperasi.

Kebijakan pemeliharaan untuk perangkat lunak AWS IoT Greengrass Inti

Perangkat lunak AWS IoT Greengrass Core v1.x memasuki fase umur yang diperpanjang pada 30 Juni 2023. Setelah tanggal ini, perangkat lunak AWS IoT Greengrass Core v1.x akan tetap dalam fase umur yang diperpanjang hingga pemberitahuan lebih lanjut.

Perangkat lunak AWS IoT Greengrass Core v2.x saat ini dalam fase rilis, dan akan tetap dalam fase rilis hingga pemberitahuan lebih lanjut. AWS IoT Greengrassterus menambahkan fitur dan penyempurnaan baru ke perangkat lunak AWS IoT Greengrass Core v2.x. Misalnya, AWS IoT Greengrass merilis dukungan Windows di v2.5.0 dari perangkat lunak AWS IoT Greengrass Core.

AWS IoT Greengrassmerilis patch keamanan dan perbaikan bug untuk semua versi minor AWS IoT Greengrass Core v2.x setidaknya selama 1 tahun setelah tanggal rilis. Untuk informasi selengkapnya, lihat Apa yang baru di AWS IoT Greengrass V2.

Jadwal fase pemeliharaan

Pada 30 Juni 2023, fase pemeliharaan berakhir untuk perangkat lunak AWS IoT Greengrass Core v1.11.x. Pada 31 Maret 2022, fase pemeliharaan berakhir untuk perangkat lunak AWS IoT Greengrass Core v1.10.x. Fase pemeliharaan berakhir untuk artefak v1.x perangkat lunak AWS IoT Greengrass Core tertentu dan fitur lebih awal dari tanggal tersebut. Untuk informasi selengkapnya, lihat Akhir jadwal pemeliharaan.

Jika Anda memiliki AWS Support rencana, fase pemeliharaan untuk perangkat lunak AWS IoT Greengrass Core v1.x tidak memengaruhi AWS Support paket Anda. Anda dapat terus membuka AWS Support tiket bahkan setelah fase pemeliharaan berakhir. Jika Anda memiliki pertanyaan atau masalah, hubungi AWS Support kontak Anda, atau ajukan pertanyaan di AWSre:Post menggunakan tag. AWS IoT Greengrass

Jadwal penghentian

Saat ini, tidak ada rencana untuk berhenti mendukung perangkat lunak AWS IoT Greengrass Core v1.x. AWS IoT Greengrass V1Titik akhir dan operasi API akan tetap tersedia hingga pemberitahuan lebih lanjut. Perangkat lunak AWS IoT Greengrass Core v1.11.6 memasuki fase umur yang diperpanjang pada 30 Juni 2023. Selama fase ini, perangkat yang menjalankan perangkat lunak AWS IoT Greengrass Core v1.x dapat terus terhubung ke AWS IoT Greengrass V1 layanan untuk beroperasi hingga pemberitahuan lebih lanjut.

Jika AWS IoT Greengrass V1 berhenti didukung di masa depan, AWS IoT Greengrass akan memberikan pemberitahuan 12 bulan sebelumnya sebelum ini terjadi. Ini akan membantu Anda merencanakan pembaruan aplikasi Anda untuk digunakan AWS IoT Greengrass V2 dan perangkat lunak AWS IoT Greengrass Core v2.x. Untuk informasi selengkapnya tentang cara memperbarui aplikasi ke V2, lihat Memindahkan dari AWS IoT Greengrass V1 ke V2.

Kebijakan Support untuk AWS Lambda fungsi pada perangkat inti Greengrass

AWS IoT Greengrassmemungkinkan Anda menjalankan AWS Lambda fungsi pada perangkat IoT. AWS Lambdamenyediakan kebijakan dukungan dan jadwal yang menentukan dukungan

untuk runtime Lambda di. AWS IoT Greengrass Setelah runtime Lambda mencapai akhir fase dukungan, AWS IoT Greengrass juga mengakhiri dukungan untuk runtime tersebut. Untuk informasi selengkapnya, lihat Kebijakan dukungan waktu aktif di Panduan Developer AWS Lambda.

Saat runtime Lambda mencapai akhir dukungan, Anda tidak dapat membuat atau memperbarui fungsi Lambda yang menggunakan runtime tersebut. Namun, Anda dapat terus menerapkan fungsi Lambda ini ke perangkat inti Greengrass dan menjalankan fungsi Lambda yang diterapkan. Kebijakan ini juga berlaku untukAWS IoT Greengrass V2.

Support kebijakan untuk AWS IoT Penguji Perangkat untuk AWS IoT Greengrass V1

AWS IoTDevice Tester (IDT) untuk AWS IoT Greengrass V1 memungkinkan Anda memvalidasi dan memenuhi syaratAWS IoT Greengrass perangkat Anda untuk dimasukkan dalam Katalog Perangkat. AWS Partner Per 4 April 2022, AWS IoT Device Tester (IDT) AWS IoT Greengrass V1 tidak lagi menghasilkan laporan kualifikasi yang ditandatangani. Anda tidak dapat lagi memenuhi syarat AWS IoT Greengrass V1 perangkat baru untuk dicantumkan di Katalog AWS Partner Perangkat melalui Program Kualifikasi AWS Perangkat. Meskipun Anda tidak dapat memenuhi syarat perangkat Greengrass V1, Anda dapat terus menggunakan IDT untuk menguji perangkat Greengrass V1 Anda. AWS IoT Greengrass V1 Kami menyarankan Anda menggunakan IDT AWS IoT Greengrass V2 untuk memenuhi syarat dan daftar perangkat Greengrass di Katalog Perangkat. AWS Partner Untuk informasi selengkapnya, lihat Support kebijakan untuk AWS IoT Penguji Perangkat untuk AWS IoT Greengrass V1.

Akhir jadwal pemeliharaan

Tabel berikut mencantumkan akhir tanggal pemeliharaan untuk artefak dan fitur AWS IoT Greengrass Core v1.x. Jika Anda memiliki pertanyaan tentang jadwal atau kebijakan pemeliharaan, hubungi AWSSupport.

Artifak atau fitur	Akhir tanggal pemeliharaan
Instalasi repositori Greengrass APT	Februari 11, 2022
Konektor Klasifikasi Citra ML	Maret 31, 2022
konektor Deteksi Objek ML	Maret 31, 2022

Artifak atau fitur	Akhir tanggal pemeliharaan
Konektor Umpan balik ML	Maret 31, 2022
AWS IoT Analyticskonektor	Maret 31, 2022
Konektor Notifikasi Twilio	Maret 31, 2022
Konektor Integrasi Splunk	Maret 31, 2022
Konektor Aliran Serial	Maret 31, 2022
ServiceNow MetricBase Konektor integrasi	Maret 31, 2022
Konektor Raspberry Pi GPIO	Maret 31, 2022
AWS IoT GreengrassPerangkat lunak inti v1.10.x	Maret 31, 2022
AWS IoT GreengrassPerangkat lunak inti v1.x gambar Docker	Juni 30, 2022
AWS IoT GreengrassPerangkat lunak inti v1.11.x	Juni 30, 2023
AWS IoT GreengrassPerangkat lunak inti v1.11.x Snap	Desember 31, 2023

Akhir pemeliharaan untuk perangkat lunak AWS IoT Greengrass Core v1.x gambar Docker

Pada tanggal 30 Juni 2022, AWS IoT Greengrass mengakhiri pemeliharaan untuk perangkat lunak AWS IoT Greengrass Core v1.x gambar Docker yang dipublikasikan ke Amazon Elastic Container Registry (Amazon ECR) Registry ECR) dan Docker Hub. Anda dapat terus mengunduh gambar Docker ini dari Amazon ECR dan Docker Hub hingga 30 Juni 2023, yaitu 1 tahun setelah pemeliharaan berakhir. Namun, gambar Docker perangkat lunak AWS IoT Greengrass Core v1.x tidak lagi menerima tambalan keamanan atau perbaikan bug setelah pemeliharaan berakhir pada 30 Juni 2022. Jika Anda menjalankan beban kerja produksi yang bergantung pada gambar Docker ini, kami sarankan Anda membuat gambar Docker Anda sendiri menggunakan Dockerfiles yang

menyediakan. AWS IoT Greengrass Untuk informasi selengkapnya, lihat AWS IoT Greengrass Perangkat lunak Docker.

Akhir pemeliharaan untuk perangkat lunak AWS IoT Greengrass inti v1.x APT repositori

Pada 11 Februari 2022, AWS IoT Greengrass mengakhiri pemeliharaan untuk opsi menginstal perangkat lunak AWS IoT Greengrass Core v1.x dari repositori APT. Repositori APT telah dihapus pada tanggal ini, sehingga Anda tidak dapat lagi menggunakan repositori APT untuk memperbarui perangkat lunak Core atau menginstal perangkat lunak AWS IoT Greengrass Core pada perangkat baru. AWS IoT Greengrass Pada perangkat tempat Anda menambahkan AWS IoT Greengrass repositori, Anda harus menghapus repositori dari daftar sumber. Kami menyarankan Anda memperbarui perangkat lunak AWS IoT Greengrass Core v1.x menggunakan file tar.

Akhir pemeliharaan untuk perangkat lunak AWS IoT Greengrass Core v1.11.x Snap

Pada 31 Desember 2023, AWS IoT Greengrass akan mengakhiri pemeliharaan untuk perangkat lunak AWS IoT Greengrass inti versi 1.11.x Snap yang diterbitkan di snapcraft.io. Perangkat yang saat ini menjalankan Snap akan terus berfungsi hingga pemberitahuan lebih lanjut. Namun, Snap AWS IoT Greengrass inti tidak akan lagi menerima tambalan keamanan atau perbaikan bug setelah pemeliharaan berakhir.

Mulai menggunakan AWS IoT Greengrass

Tutorial Memulai ini mencakup beberapa modul yang dirancang untuk menunjukkan kepada Anda dasar-dasar AWS IoT Greengrass dan membantu Anda mulai menggunakan AWS IoT Greengrass. Tutorial ini meliputi konsep-konsep dasar, seperti:

- Mengonfigurasi core AWS IoT Greengrass dan grup.
- Proses deployment untuk menjalankan fungsi AWS Lambda di edge.
- Menghubungkan AWS IoT perangkat, yang disebut perangkat klien, ke AWS IoT Greengrass inti.
- Membuat langganan untuk memungkinkan komunikasi MQTT antara fungsi Lambda lokal, perangkat klien, dan. AWS IoT

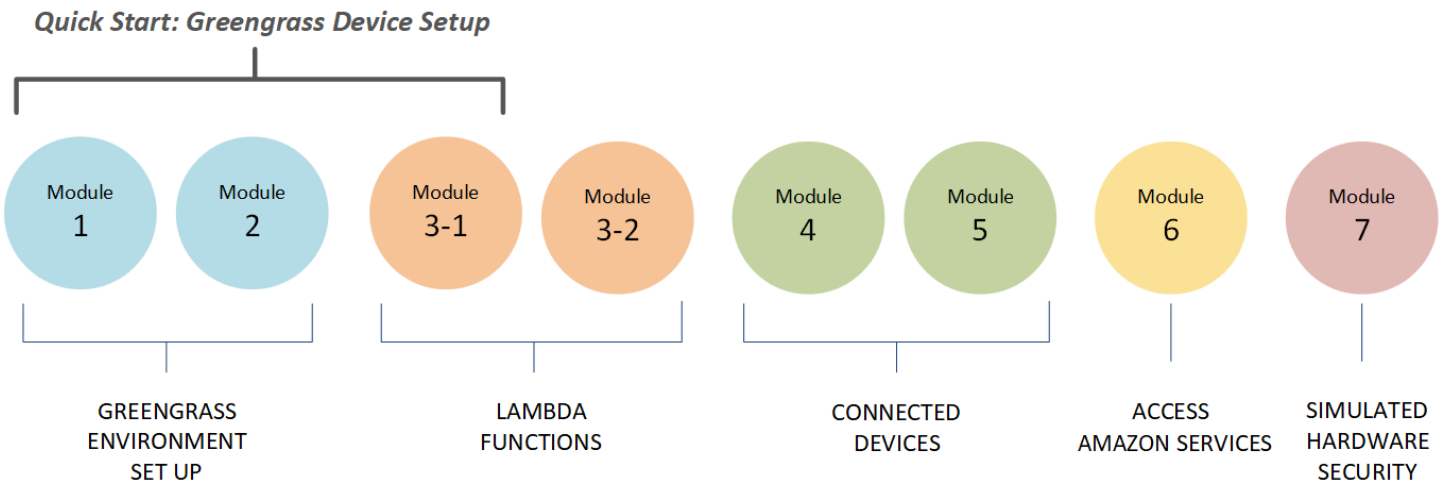
Pilih cara memulai dengan AWS IoT Greengrass

Anda dapat memilih cara menggunakan tutorial ini untuk mengatur perangkat core Anda:

- Jalankan penyiapan perangkat Greengrass pada perangkat core Anda, yang akan membawa Anda dari penginstalan dependensi AWS IoT Greengrass untuk menguji fungsi Hello World Lambda dalam hitungan menit. Skrip ini mereproduksi langkah-langkah dari Modul 1 sampai Modul 3-1.

  - atau -

- Ikuti langkah-langkah di Modul 1 hingga Modul 3-1 untuk mempelajari persyaratan dan proses Greengrass lebih dekat. Langkah-langkah ini mengatur perangkat core Anda, membuat dan mengonfigurasi grup Greengrass yang berisi fungsi Hello World Lambda, dan menjalankan grup Greengrass Anda. Biasanya, ini membutuhkan waktu satu atau dua jam untuk menyelesaikannya.



Mulai Cepat

Penyiapan perangkat Greengrass mengonfigurasi perangkat core dan sumber daya Greengrass Anda. Skrip:

- Menginstal AWS IoT Greengrass dependensi.
- Mengunduh sertifikat CA root dan sertifikat perangkat core dan kunci.
- Mengunduh, menginstal, dan mengonfigurasi AWS IoT Greengrass perangkat lunak Core di perangkat Anda.
- Memulai proses Greengrass daemon pada perangkat core.
- Membuat atau memperbarui peran layanan Greengrass, jika diperlukan.
- Membuat grup Greengrass dan core Greengrass.
- (Opsional) Membuat fungsi Hello World Lambda, langganan, dan konfigurasi logging lokal.
- (Opsional) Men-deploy grup Greengrass.

Modul 1 dan 2

Modul 1 dan Modul 2 menjelaskan cara mengatur lingkungan Anda. (Atau, gunakan penyiapan perangkat Greengrass untuk menjalankan modul ini untuk Anda.)

- Konfigurasikan perangkat core Anda untuk Greengrass.
- Jalankan skrip pemeriksa depedensi.
- Buat grup Greengrass dan core Greengrass.
- Unduh dan instal perangkat lunak Core AWS IoT Greengrass terbaru dari file tar.gz.

- Mulai proses Greengrass daemon pada core.

> **Note**
>
> AWS IoT Greengrass juga menyediakan opsi lain untuk menginstal AWS IoT Greengrass perangkat lunak Core, termasuk `apt` penginstalan pada platform Debian yang didukung. Untuk informasi selengkapnya, lihat the section called “Instal AWS IoT Greengrass perangkat lunak Core”.

Modul 3-1 dan 3-2

Modul 3-1 dan Modul 3-2 menjelaskan cara menggunakan fungsi Lambda lokal. (Atau, gunakan penyiapan perangkat Greengrass untuk menjalankan Modul 3-1 untuk Anda.)

- Buat fungsi Hello World Lambda di AWS Lambda.
- Tambahkan fungsi Lambda ke grup Greengrass Anda.
- Buat langganan yang mengizinkan komunikasi MQTT antara fungsi Lambda dan AWS IoT.
- Konfigurasikan logging lokal untuk komponen sistem Greengrass dan fungsi Lambda.
- Men-deploy grup Greengrass yang berisi fungsi Lambda Anda dan langganan.
- Kirim pesan dari fungsi Lambda lokal ke AWS IoT.
- Minta fungsi Lambda lokal dari AWS IoT.
- Uji fungsi on-demand dan long-lived.

Modul 4 dan 5

Modul 4 menunjukkan bagaimana perangkat klien terhubung ke inti dan berkomunikasi satu sama lain.

Modul 5 menunjukkan bagaimana perangkat klien dapat menggunakan bayangan untuk mengontrol status.

- Pendaftaran dan ketentuan AWS IoT perangkat (diwakili oleh terminal baris perintah).
- Instal AWS IoT Device SDK untuk Python. Ini digunakan oleh perangkat klien untuk menemukan inti Greengrass.
- Tambahkan perangkat klien ke grup Greengrass Anda.
- Buat langganan yang mengizinkan komunikasi MQTT.

- Terapkan grup Greengrass yang berisi perangkat klien Anda.
- Uji device-to-device komunikasi.
- Uji memperbarui status keadaan.

Modul 6

Modul 6 menunjukkan cara fungsi Lambda dapat mengakses AWS Cloud.

- Buat peran grup Greengrass yang mengizinkan akses ke sumber daya Amazon DynamoDB.
- Tambahkan sebuah fungsi Lambda ke grup Greengrass Anda. Fungsi ini menggunakan AWS SDK for Python untuk berinteraksi dengan DynamoDB.
- Buat langganan yang mengizinkan komunikasi MQTT.
- Tes interaksi dengan DynamoDB.

Modul 7

Modul 7 menunjukkan cara mengonfigurasi modul keamanan perangkat keras simulasi (HSM) untuk digunakan dengan core Greengrass.

> **Important**
>
> Modul lanjutan ini disediakan hanya untuk eksperimen dan pengujian awal. Ini bukan untuk penggunaan produksi dalam bentuk apa pun.

- Instal dan konfigurasikan perangkat lunak berbasis HSM dan kunci privat.
- Konfigurasikan core Greengrass menggunakan keamanan perangkat keras.
- Uji konfigurasi keamanan perangkat keras.

Persyaratan

Untuk menyelesaikan tutorial ini, Anda perlu berikut:

- Mac, Windows PC, atau sistem UNIX-like.
- Sesi Akun AWS. Jika Anda tidak memilikinya, lihat the section called “Buat Akun AWS”.
- Penggunaan AWS Wilayah yang mendukung AWS IoT Greengrass. Untuk daftar wilayah yang didukungAWS IoT Greengrass, lihat AWStitik akhir dan kuota di. Referensi Umum AWS

> **Note**
>
> Buat catatan Anda Wilayah AWS dan pastikan bahwa itu secara konsisten digunakan di seluruh tutorial ini. Jika Anda mengganti perangkat Wilayah AWS selama tutorial, Anda mungkin mengalami masalah dalam menyelesaikan langkah-langkahnya.

- A Raspberry Pi 4 Model B, atau Raspberry Pi 3 Model B/B+, dengan kartu microSD 8 GB, atau contoh Amazon EC2. Karena AWS IoT Greengrass idealnya harus digunakan dengan perangkat keras fisik, kami sarankan Anda menggunakan Raspberry Pi.

  > **Note**
  >
  > Jalankan perintah berikut untuk mendapatkan model Anda Raspberry Pi:
  >
  > ```
  > cat /proc/cpuinfo
  > ```
  >
  > Dekat bagian bawah daftar, buat catatan dari nilai `Revision` atribut lalu konsultasi Pi yang mana yang saya punya? tabel. Sebagai contoh, jika nilai `Revision` adalah `a02082`, tabel menunjukkan Pi adalah 3 Model B.
  > Jalankan perintah berikut untuk menentukan arsitektur dari Anda Raspberry Pi:
  >
  > ```
  > uname -m
  > ```
  >
  > Untuk tutorial ini, hasilnya harus lebih besar dari atau sama dengan `armv71`.

- Kebiasaan dasar dengan Python.

Meskipun tutorial ini dimaksudkan untuk menjalankan AWS IoT Greengrass pada Raspberry Pi, AWS IoT Greengrass juga mendukung platform lain. Untuk informasi selengkapnya, lihat the section called “Platform dan persyaratan yang didukung”.

Buat Akun AWS

Jika Anda tidak memiliki Akun AWS, ikuti langkah-langkah ini untuk membuat dan mengaktifkan Akun AWS:

Mendaftar Akun AWS

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar Akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.
2. Ikuti petunjuk secara online.

   Anda akan diminta untuk menerima panggilan telepon dan memasukkan kode verifikasi pada keypad telepon sebagai bagian dari prosedur pendaftaran.

   Saat Anda mendaftar Akun AWS, Pengguna root akun AWS akan dibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya dalam akun. Sebagai praktik terbaik keamanan, tetapkan akses administratif ke pengguna administratif, dan hanya gunakan pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

AWS akan mengirimkan email konfirmasi kepada Anda setelah proses pendaftaran selesai. Anda dapat melihat aktivitas akun saat ini dan mengelola akun dengan mengunjungi https://aws.amazon.com/ dan memilih Akun Saya.

Membuat pengguna administratif

Setelah mendaftar Akun AWS, amankan Pengguna root akun AWS, aktifkan AWS IAM Identity Center, dan buat sebuah pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Mengamankan Pengguna root akun AWS Anda

1. Masuk ke AWS Management Console sebagai pemilik akun dengan memilih Pengguna root dan memasukkan alamat email Akun AWS Anda. Di halaman berikutnya, masukkan kata sandi Anda.

   Untuk bantuan masuk menggunakan pengguna root, lihat Masuk sebagai pengguna root dalam Panduan Pengguna AWS Sign-In.
2. Aktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda.

   Untuk petunjuknya, silakan lihat Mengaktifkan perangkat MFA virtual untuk pengguna root Akun AWS Anda (konsol) dalam Panduan Pengguna IAM.

Membuat pengguna administratif

1. Aktifkan Pusat Identitas IAM.

   Untuk mendapatkan petunjuk, silakan lihat Mengaktifkan AWS IAM Identity Center di Panduan Pengguna AWS IAM Identity Center.

2. Di Pusat Identitas IAM, berikan akses administratif ke sebuah pengguna administratif.

   Untuk mendapatkan tutorial tentang menggunakan Direktori Pusat Identitas IAM sebagai sumber identitas Anda, silakan lihat Mengonfigurasi akses pengguna dengan Direktori Pusat Identitas IAM default di Panduan Pengguna AWS IAM Identity Center.

Masuk sebagai pengguna administratif

- Untuk masuk dengan pengguna Pusat Identitas IAM, gunakan URL masuk yang dikirim ke alamat email Anda saat Anda membuat pengguna Pusat Identitas IAM.

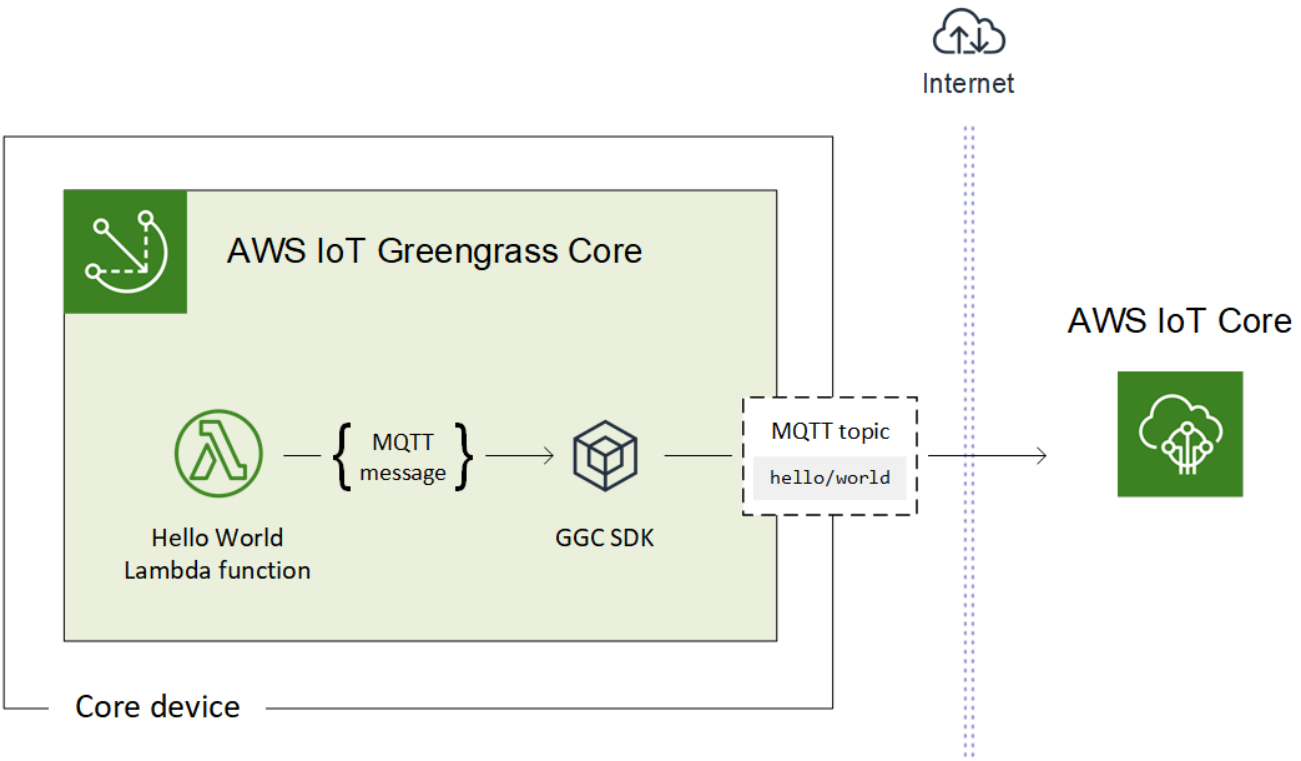
  Untuk bantuan masuk menggunakan pengguna Pusat Identitas IAM, lihat Masuk ke portal akses AWS dalam Panduan Pengguna AWS Sign-In.

> **Important**
>
> Untuk tutorial ini, kita berasumsi bahwa akun pengguna IAM Anda memiliki izin akses administrator.

Quick start: penyiapan perangkat Greengrass

Penyapan perangkat Greengrass adalah skrip yang mengatur perangkat core Anda dalam hitungan menit, sehingga Anda dapat mulai menggunakan AWS IoT Greengrass. Gunakan skrip ini untuk:

1. Konfigurasikan perangkat Anda dan instal AWS IoT Greengrass perangkat lunak Core.
2. Konfigurasikan sumber daya berbasis cloud.
3. Opsional terapkan grup Greengrass dengan fungsi Hello World Lambda yang mengirimkan pesan MQTT ke AWS IoT dari AWS IoT Greengrass core. Ini mengatur lingkungan Greengrass yang ditunjukkan dalam diagram berikut.



Persyaratan

Penyiapan perangkat Greengrass memiliki persyaratan sebagai berikut:

- Perangkat core Anda harus menggunakan platform yang didukung. Perangkat harus memiliki manajer paket yang sesuai diinstal: `apt`, `yum`, atau `opkg`.

- Pengguna Linux yang menjalankan skrip harus memiliki izin untuk menjalankan sebagai `sudo`.

- Anda harus menyediakan Akun AWS kredensial. Untuk informasi selengkapnya, lihat the section called “Sediakan Akun AWS kredensial”.

> **Note**
>
> Penyiapan perangkat Greengrass menginstal versi terbaru dari AWS IoT Greengrass perangkat Core pada perangkat. Dengan menginstal AWS IoT Greengrass perangkat Core, Anda menyetujui Perjanjian Lisensi Software Greengrass Core.

Jalankan penyiapan perangkat Greengrass

Anda dapat menjalankan penyiapan perangkat Greengrass hanya dalam beberapa langkah. Setelah Anda menyediakan Akun AWS kredensial, skrip ketentuan perangkat core Greengrass Anda dan terapkan grup Greengrass dalam menit. Jalankan perintah berikut di jendela terminal pada perangkat target.

> **Note**
>
> Langkah-langkah ini menunjukkan cara menjalankan skrip dalam mode interaktif, yang meminta Anda untuk memasukkan atau menerima setiap nilai input. Untuk informasi tentang cara menjalankan skrip secara senyap, lihat the section called “Jalankan penyiapan perangkat Greengrass dalam mode diam”.

1. Sediakan kredensial Anda. Dalam prosedur ini, kami menganggap Anda menyediakan kredensial keamanan sementara sebagai variabel lingkungan.

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

   > **Note**
   >
   > Jika Anda menjalankan penyiapan perangkat Greengrass pada Raspbian atau OpenWrt platform, buat salinan perintah ini. Anda harus menyediakannya lagi setelah Anda me-reboot perangkat.

2. Unduh dan mulai skrip. Anda dapat menggunakan `wget` atau `curl` untuk mengunduh skrip.

   `wget`:

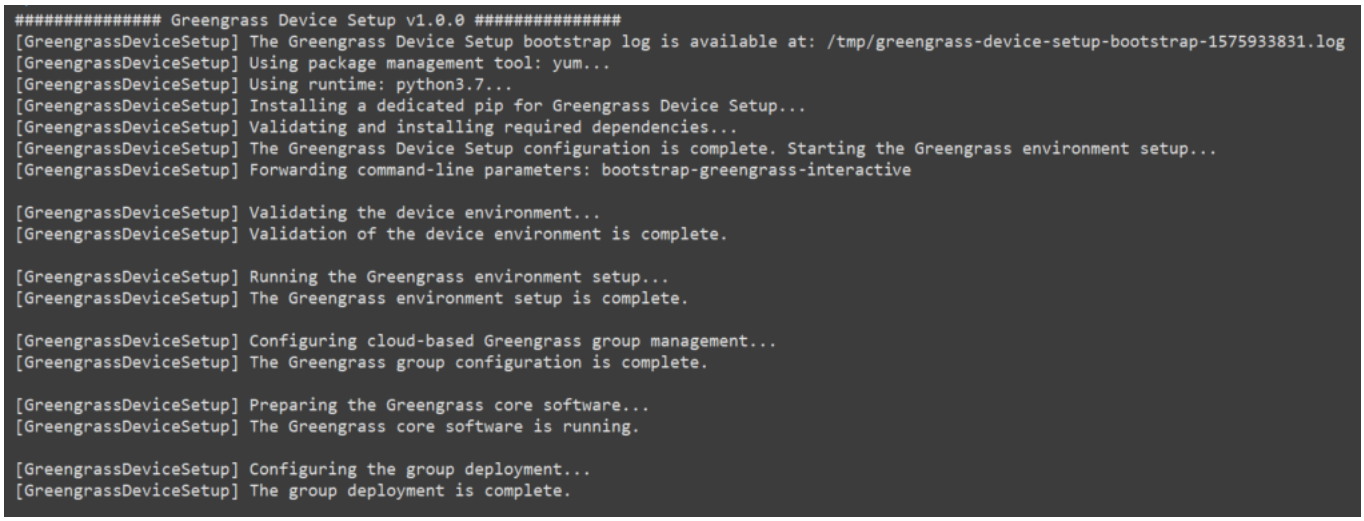
   ```
   wget -q -O ./gg-device-setup-latest.sh https://d1onfpft10uf5o.cloudfront.net/
   greengrass-device-setup/downloads/gg-device-setup-latest.sh && chmod +x ./
   gg-device-setup-latest.sh && sudo -E ./gg-device-setup-latest.sh bootstrap-
   greengrass-interactive
   ```

`curl`:

```
curl https://d1onfpft10uf5o.cloudfront.net/greengrass-device-setup/downloads/
gg-device-setup-latest.sh > gg-device-setup-latest.sh && chmod +x ./gg-device-
setup-latest.sh && sudo -E ./gg-device-setup-latest.sh bootstrap-greengrass-
interactive
```

3. Lanjutkan melalui perintah prompt untuk nilai input. Anda dapat menekan tombol Masukkan untuk menggunakan nilai default atau ketik nilai kustom lalu tekan Masukkan.

   Skrip menulis pesan status ke terminal yang mirip dengan berikut ini.



4. Jika perangkat core Anda menjalankan Raspbian atau OpenWrt reboot perangkat ketika diminta, sediakan kredenial Anda, lalu restart skrip.

   a. Ketika diminta untuk reboot perangkat, jalankan salah satu perintah berikut.

      Untuk platform Raspbian:

      ```
      sudo reboot
      ```

      Untuk OpenWrt platform:

      ```
      reboot
      ```

b. Setelah perangkat reboot, buka terminal dan sediakan kredensial Anda sebagai variabel lingkungan.

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

c. Restart skrip.

```
sudo -E ./gg-device-setup-latest.sh bootstrap-greengrass-interactive
```

d. Ketika diminta apakah akan menggunakan nilai input dari sesi sebelumnya atau memulai penginstalan baru, masukkan `yes` untuk menggunakan kembali nilai input Anda.

> **Note**
>
> Pada platform yang memerlukan reboot, nilai input Anda dari sesi sebelumnya, tidak termasuk kredensial, untuk sementara disimpan di `GreengrassDeviceSetup.config.info` file.

Setelah penyiapan selesai, terminal akan menampilkan pesan status sukses yang mirip dengan berikut ini.

```
================================================================================

Your device is running the Greengrass core software.
Your Greengrass group and Hello World Lambda function were deployed to the core device.


Setup information:

Device info: Linux-4.14.152-127.182.amzn2.x86_64-x86_64-with-glibc2.2.5
Greengrass core software location: /
Installed Greengrass core software version: 1.10.0
Greengrass core: arn:aws:iot:us-west-2:012345678910:thing/GreengrassDeviceSetup_Core_d46a0ea4-18ae-4376-8f44-4a504cdea608
Greengrass core IoT certificate: arn:aws:iot:us-west-2:012345678910:cert/23fbf0f4b6a5ea369f2b97f1a1b558180a240faa8e059ce19dc58f4a4c0d3b77
Greengrass core IoT certificate location: /greengrass/certs/23fbf0f4b6.cert.pem
Greengrass core IoT key location: /greengrass/certs/23fbf0f4b6.private.key
Deployed Greengrass group name: GreengrassDeviceSetup_Group_ee70f777-9af0-43b6-8612-a18b418e8b4a
Deployed Greengrass group ID: 6f5c8410-f3a6-43a2-acf3-33158e10fb8e
Deployed Greengrass group version: arn:aws:greengrass:us-west-2:012345678910:/greengrass/groups/6f5c8410-f3a6-43a2-acf3-33158e10fb8e/versi
Greengrass service role: arn:aws:iam::012345678910:role/GreengrassServiceRole_mui1v
GreengrassDeviceSetup log location: GreengrassDeviceSetup-20191209-232356.log
Deployed hello-world Lambda function: arn:aws:lambda:us-west-2:012345678910:function:Greengrass_HelloWorld_uNTf2:1
Hello-world subscriber topic: hello/world

You can now use the AWS IoT Console to subscribe
to the 'hello/world' topic to receive messages published from your
Greengrass core.

================================================================================
```

5. Tinjau grup Greengrass baru yang skrip mengonfigurasi menggunakan nilai-nilai input yang Anda sediakan.

a. Masuk ke AWS Management Console di komputer Anda dan buka AWS IoT konsol.

> **Note**
>
> Pastikan Wilayah AWS yang dipilih di konsol sama dengan yang Anda gunakan untuk mengonfigurasi lingkungan Greengrass Anda. Secara default, Wilayah adalah US West (Oregon).

b. Di panel navigasi, luaskan perangkat Greengrass, kemudian pilih Grup (V1) untuk menempatkan grup yang baru dibuat.

6. Jika Anda menyertakan fungsi Lambda, penyiapan perangkat Greengrass men-deploy grup Greengrass ke perangkat core Anda. Untuk menguji fungsi Lambda, atau untuk informasi tentang cara menghapus fungsi Lambda dari grup, lanjutkan ke the section called “Verifikasi fungsi Lambda berjalan pada perangkat core” di Modul 3-1 dari tutorial Memulai.

> **Note**
>
> Pastikan Wilayah AWS yang dipilih di konsol sama dengan yang Anda gunakan untuk mengonfigurasi lingkungan Greengrass Anda. Secara default, Wilayah adalah US West (Oregon).

Jika Anda tidak menyertakan fungsi Lambda Hello World, Anda dapat membuat fungsi Lambda Anda sendiri atau mencoba fitur Greengrass lainnya. Sebagai contoh, Anda dapat menambahkan konektor Deployment aplikasi Docker ke grup Anda dan menggunakannya untuk men-deploy kontainer Docker ke perangkat core Anda.

Memecahkan masalah

Informasi berikut dapat membantu Anda menyelesaikan masalah dengan AWS IoT Greengrass penyiapan perangkat.

Kesalahan: Python (python3.7) tidak ditemukan. Mencoba untuk menginstalnya...

Solusi: Anda mungkin melihat kesalahan ini ketika bekerja dengan Amazon EC2 contoh. Kesalahan ini terjadi ketika Python tidak diinstal di folder `/usr/bin/python3.7` ini. Untuk mengatasi kesalahan ini, pindahkan Python dalam direktori yang benar setelah menginstalnya:

```
sudo ln -s /usr/local/bin/python3.7 /usr/bin/python3.7
```

Pemecahan masalah tambahan

Untuk memecahkan masalah tambahan dengan AWS IoT Greengrass penyiapan perangkat, Anda dapat mencari informasi debug dalam berkas log:

- Untuk masalah dengan konfigurasi penyiapan perangkat Greengrass, periksa `/tmp/greengrass-device-setup-bootstrap-`*`epoch-timestamp`*`.log` file.
- Untuk masalah dengan grup Greengrass atau penyiapan lingkungan core, periksa `GreengrassDeviceSetup-`*`date`*`-`*`time`*`.log` file dalam direktori yang sama seperti `gg-device-setup-latest.sh` atau di lokasi yang Anda tentukan.

Untuk bantuan pemecahan masalah lainnya, lihatMemecahkan masalah atau periksa AWS IoT Greengrasstag diAWS Re:post.

Pilihan konfigurasi penyiapan perangkat Greengrass

Anda mengonfigurasi penyiapan perangkat Greengrass untuk mengakses AWS sumber daya dan mengatur lingkungan Greengrass Anda.

Sediakan Akun AWS kredensial

Penyiapan perangkat Greengrass menggunakan Akun AWS kredensial untuk mengakses AWS sumber daya. Itu mendukung kredensials jangka panjang untuk pengguna IAM atau kredensial keamanan sementara dari IAM role.

Pertama, dapatkan kredensial Anda.

- Untuk menggunakan kredensial jangka panjang, sediakan access key ID dan secret access key untuk pengguna IAM Anda. Untuk informasi tentang cara membuat access key untuk kredensial jangka panjang, lihat Mengelola access key untuk pengguna IAM di Panduan Pengguna IAM.

- Untuk menggunakan kredensial keamanan sementara (disarankan), sediakan access key ID, secret access key, dan token sesi dari IAM role yang diasumsikan. Untuk informasi tentang penggalian kredensial keamanan sementara dari AWS STS `assume-role` perintah, lihat Menggunakan kredensial keamanan sementara dengan AWS CLI di Panduan Pengguna IAM.

> **Note**
>
> Untuk tujuan tutorial ini, kita berasumsi bahwa pengguna IAM atau IAM role memiliki izin akses administrator.

Kemudian, sediakan kredensial Anda untuk penyiapan perangkat Greengrass dengan salah satu dari dua cara berikut:

- Sebagai variabel lingkungan. Atur `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, dan `AWS_SESSION_TOKEN` (jika diperlukan) variabel lingkungan sebelum Anda mulai skrip, seperti yang ditunjukkan pada langkah 1 dari the section called “Jalankan penyiapan perangkat Greengrass”.

- Sebagai nilai masukan. Masukkan access key ID Anda, secret access key, dan nilai token sesi (jika diperlukan) secara langsung di terminal setelah Anda memulai skrip.

Penyiapan perangkat Greengrass tidak menyimpan atau menyimpan kredensial Anda.

Sediakan nilai input

Dalam mode interaktif, pengaturan perangkat Greengrass meminta Anda untuk nilai input. Anda dapat menekan tombol Masukkan untuk menggunakan nilai default atau ketik nilai kustom lalu tekan Masukkan. Dalam mode diam, Anda menyediakan nilai input setelah Anda memulai skrip.

Nilai input

AWSaccess key ID

Access key ID dari kredensial keamanan jangka panjang atau sementara. Tentukan opsi ini sebagai nilai input hanya jika Anda tidak menyediakan kredensial Anda sebagai variabel

lingkungan. Untuk informasi selengkapnya, lihat the section called “Sediakan Akun AWS kredensial”.

Nama opsi untuk mode diam: `--aws-access-key-id`

AWSsecret access key

Secret access key dari kredensial keamanan jangka panjang atau sementara. Tentukan opsi ini sebagai nilai input hanya jika Anda tidak menyediakan kredensial Anda sebagai variabel lingkungan. Untuk informasi selengkapnya, lihat the section called “Sediakan Akun AWS kredensial”.

Nama opsi untuk mode diam: `--aws-secret-access-key`

AWStoken sesi

Token sesi dari kredensial keamanan sementara. Tentukan opsi ini sebagai nilai input hanya jika Anda tidak menyediakan kredensial Anda sebagai variabel lingkungan. Untuk informasi selengkapnya, lihat the section called “Sediakan Akun AWS kredensial”.

Nama opsi untuk mode diam: `--aws-session-token`

Wilayah AWS

Di mana Wilayah AWS Anda ingin membuat grup Greengrass. Untuk daftarWilayah AWS s didukung, lihat AWS IoT Greengrassdi Referensi Umum Amazon Web Services.

Nilai default: `us-west-2`

Nama opsi untuk mode diam: `--region`

Nama grup

Nama untuk grup Greengrass.

Nilai default: `GreengrassDeviceSetup_Group_`*`guid`*

Nama opsi untuk mode diam: `--group-name`

Nama inti

Nama untuk core Greengrass. Core adalah AWS IoT perangkat (hal) yang menjalankan AWS IoT Greengrass perangkat lunak Core. Core ditambahkan ke AWS IoT registri dan grup Greengrass. Jika Anda menyediakan nama, itu harus unik dalam Akun AWS dan Wilayah AWS.

Nilai default: `GreengrassDeviceSetup_Core_`*`guid`*

Nama opsi untuk mode diam: `--core-name`

AWS IoT GreengrassJalur penginstalan perangkat lunak core

Lokasi dalam sistem file perangkat di mana Anda ingin menginstal AWS IoT Greengrass perangkat lunak Core.

Nilai default: `/`

Nama opsi untuk mode diam: `--ggc-root-path`

Fungsi Lambda

Menunjukkan apakah akan menyertakan fungsi Hello World Lambda dalam grup Greengrass. Fungsi menerbitkan pesan MQTT ke `hello/world` topik setiap lima detik.

Skrip membuat dan menerbitkan fungsi Lambda yang ditetapkan pengguna ini di AWS Lambda dan menambahkannya ke grup Greengrass Anda. Skrip juga membuat langganan dalam grup yang mengizinkan fungsi mengirim pesan MQTT ke AWS IoT.

> **Note**
>
> Ini adalah Python 3.7 fungsi Lambda. Jika Python 3.7 tidak diinstal pada perangkat dan skrip tidak dapat menginstalnya, skrip mencetak pesan kesalahan di terminal. Untuk menyertakan fungsi Lambda dalam grup, Anda harus menginstal Python 3.7 secara manual dan restart skrip. Untuk membuat grup Greengrass tanpa fungsi Lambda, restart skrip dan masukkan no ketika diminta untuk menyertakan fungsi.

Nilai default: no

Nama opsi untuk mode diam: `--hello-world-lambda` - Opsi ini tidak mengambil nilai. Sertakan dalam perintah Anda jika Anda ingin membuat fungsi.

Batas waktu penyebaran

Jumlah detik sebelum pengaturan perangkat Greengrass berhenti memeriksa status deployment grup Greengrass. Ini hanya digunakan ketika grup menyertakan fungsi Hello World Lambda. Jika tidak, grup tidak di-deploy.

Waktu deployment tergantung pada kecepatan jaringan Anda. Untuk kecepatan jaringan yang lambat, Anda dapat meningkatkan nilai ini.

Nilai default: `180`

Nama opsi untuk mode diam: `--deployment-timeout`

Jalur log

Lokasi berkas log yang berisi informasi tentang Greengrass grup dan operasi penyiapan core. Gunakan log ini untuk memecahkan masalah deployment dan masalah lainnya dengan grup Greengrass dan penyiapan Core.

Nilai default: `./`

Nama opsi untuk mode diam: `--log-path`

Verbositas

Mengindikasikan apakah akan mencetak informasi log terperinci di terminal sementara skrip berjalan. Anda dapat menggunakan informasi ini untuk memecahkan masalah penyiapan perangkat.

Nilai default: no

Nama opsi untuk mode diam: `--verbose` - Opsi ini tidak mengambil nilai. Sertakan dalam perintah Anda jika Anda ingin mencetak informasi log terperinci.

Jalankan penyiapan perangkat Greengrass dalam mode diam

Anda dapat menjalankan penyiapan perangkat Greengrass dalam mode diam sehingga skrip tidak meminta Anda untuk nilai-nilai apa pun. Untuk menjalankan dalam mode senyap, tentukan `bootstrap-greengrass` dan mode nilai input setelah Anda memulai skrip. Anda dapat menghilangkan nilai input jika Anda ingin menggunakan default mereka.

Prosedur ini tergantung pada apakah Anda menyediakan Akun AWS kredensial sebagai variabel lingkungan sebelum Anda memulai skrip, atau sebagai nilai input setelah Anda memulai skrip.

Sediakan kredensial sebagai variabel lingkungan

1. Sediakan kredensial Anda sebagai variabel lingkungan. Contoh berikut ekspor kredensial sementara, yang meliputi token sesi.

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

> **Note**
>
> Jika Anda menjalankan penyiapan perangkat Greengrass pada Raspbian atau OpenWrt platform, buat salinan perintah ini. Anda harus menyediakannya lagi setelah Anda me-reboot perangkat.

2. Unduh dan mulai skrip. Sediakan nilai input yang dibutuhkan. Misalnya:

   - Untuk menggunakan semua nilai default:

   ```
   wget -q -O ./gg-device-setup-latest.sh https://d1onfpft10uf5o.cloudfront.net/
   greengrass-device-setup/downloads/gg-device-setup-latest.sh && chmod +x ./
   gg-device-setup-latest.sh && sudo -E ./gg-device-setup-latest.sh bootstrap-
   greengrass
   ```

   - Untuk menentukan nilai kustom:

   ```
   wget -q -O ./gg-device-setup-latest.sh https://d1onfpft10uf5o.cloudfront.net/
   greengrass-device-setup/downloads/gg-device-setup-latest.sh && chmod +x ./
   gg-device-setup-latest.sh && sudo -E ./gg-device-setup-latest.sh bootstrap-
   greengrass
   --region us-east-1
   --group-name Custom_Group_Name
   --core-name Custom_Core_Name
   --ggc-root-path /custom/ggc/root/path
   --deployment-timeout 300
   --log-path /customized/log/path
   --hello-world-lambda
   --verbose
   ```

> **Note**
>
> Untuk menggunakan `curl` agar mengunduh skrip, ganti `wget -q -O` dengan `curl` dalam perintah.

3. Jika perangkat core Anda menjalankan Raspbian atau OpenWrt reboot perangkat ketika diminta, sediakan kredenial Anda, lalu restart skrip.

   a. Ketika diminta untuk reboot perangkat, jalankan salah satu perintah berikut.

Untuk platform Raspbian:

```
sudo reboot
```

Untuk OpenWrt platform:

```
reboot
```

b. Setelah perangkat reboot, buka terminal dan sediakan kredensial Anda sebagai variabel lingkungan.

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

c. Restart skrip.

```
sudo -E ./gg-device-setup-latest.sh bootstrap-greengrass
```

d. Ketika diminta apakah akan menggunakan nilai input dari sesi sebelumnya atau memulai penginstalan baru, masukkan `yes` untuk menggunakan kembali nilai input Anda.

> **Note**
>
> Pada platform yang memerlukan reboot, nilai input Anda dari sesi sebelumnya, tidak termasuk kredensial, untuk sementara disimpan di `GreengrassDeviceSetup.config.info` file.

Setelah penyiapan selesai, terminal akan menampilkan pesan status sukses yang mirip dengan berikut ini.

```
================================================================================

Your device is running the Greengrass core software.
Your Greengrass group and Hello World Lambda function were deployed to the core device.


Setup information:

Device info: Linux-4.14.152-127.182.amzn2.x86_64-x86_64-with-glibc2.2.5
Greengrass core software location: /
Installed Greengrass core software version: 1.10.0
Greengrass core: arn:aws:iot:us-west-2:012345678910:thing/GreengrassDeviceSetup_Core_d46a0ea4-18ae-4376-8f44-4a504cdea608
Greengrass core IoT certificate: arn:aws:iot:us-west-2:012345678910:cert/23fbf0f4b6a5ea369f2b97f1a1b558180a240faa8e059ce19dc58f4a4c0d3b77
Greengrass core IoT certificate location: /greengrass/certs/23fbf0f4b6.cert.pem
Greengrass core IoT key location: /greengrass/certs/23fbf0f4b6.private.key
Deployed Greengrass group name: GreengrassDeviceSetup_Group_ee70f777-9af0-43b6-8612-a18b418e8b4a
Deployed Greengrass group ID: 6f5c8410-f3a6-43a2-acf3-33158e10fb8e
Deployed Greengrass group version: arn:aws:greengrass:us-west-2:012345678910:/greengrass/groups/6f5c8410-f3a6-43a2-acf3-33158e10fb8e/versi
Greengrass service role: arn:aws:iam::012345678910:role/GreengrassServiceRole_mui1v
GreengrassDeviceSetup log location: GreengrassDeviceSetup-20191209-232356.log
Deployed hello-world Lambda function: arn:aws:lambda:us-west-2:012345678910:function:Greengrass_HelloWorld_uNTf2:1
Hello-world subscriber topic: hello/world

You can now use the AWS IoT Console to subscribe
to the 'hello/world' topic to receive messages published from your
Greengrass core.

================================================================================
```

4. Jika Anda menyertakan fungsi Lambda, penyiapan perangkat Greengrass men-deploy grup Greengrass ke perangkat core Anda. Untuk menguji fungsi Lambda, atau untuk informasi tentang cara menghapus fungsi Lambda dari grup, lanjutkan ke the section called “Verifikasi fungsi Lambda berjalan pada perangkat core” di Modul 3-1 dari tutorial Memulai.

   > **Note**
   >
   > Pastikan Wilayah AWS yang dipilih di konsol sama dengan yang Anda gunakan untuk mengonfigurasi lingkungan Greengrass Anda. Secara default, Wilayah adalah US West (Oregon).

   Jika Anda tidak menyertakan fungsi Lambda Hello World, Anda dapat membuat fungsi Lambda Anda sendiri atau mencoba fitur Greengrass lainnya. Sebagai contoh, Anda dapat menambahkan konektor Deployment aplikasi Docker ke grup Anda dan menggunakannya untuk men-deploy kontainer Docker ke perangkat Core Anda.

Sediakan kredensial sebagai nilai input

1. Unduh dan mulai skrip. Sediakan kredensial Anda dan nilai input lainnya yang ingin Anda tentukan. Contoh berikut menunjukkan cara menyediakan kredensial sementara, yang mencakup token sesi.

- Untuk menggunakan semua nilai default:

```
wget -q -O ./gg-device-setup-latest.sh https://d1onfpft10uf5o.cloudfront.net/
greengrass-device-setup/downloads/gg-device-setup-latest.sh && chmod +x ./
gg-device-setup-latest.sh && sudo -E ./gg-device-setup-latest.sh bootstrap-
greengrass
--aws-access-key-id AKIAIOSFODNN7EXAMPLE
--aws-secret-access-key wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
--aws-session-token AQoDYXdzEJr1K...o5OytwEXAMPLE=
```

- Untuk menentukan nilai kustom:

```
wget -q -O ./gg-device-setup-latest.sh https://d1onfpft10uf5o.cloudfront.net/
greengrass-device-setup/downloads/gg-device-setup-latest.sh && chmod +x ./
gg-device-setup-latest.sh && sudo -E ./gg-device-setup-latest.sh bootstrap-
greengrass
--aws-access-key-id AKIAIOSFODNN7EXAMPLE
--aws-secret-access-key wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
--aws-session-token AQoDYXdzEJr1K...o5OytwEXAMPLE=
--region us-east-1
--group-name Custom_Group_Name
--core-name Custom_Core_Name
--ggc-root-path /custom/ggc/root/path
--deployment-timeout 300
--log-path /customized/log/path
--hello-world-lambda
--verbose
```

> **Note**
>
> Jika Anda menjalankan penyiapan perangkat Greengrass pada Raspbian atau OpenWrt platform, buat salinan kredenial Anda. Anda harus menyediakannya lagi setelah Anda me-reboot perangkat.
>
> Untuk menggunakan `curl` agar mengunduh skrip, ganti `wget -q -O` dengan `curl` dalam perintah.

2. Jika perangkat core Anda menjalankan Raspbian atau OpenWrt reboot perangkat ketika diminta, sediakan kredenial Anda, lalu restart skrip.

a. Ketika diminta untuk reboot perangkat, jalankan salah satu perintah berikut.

   Untuk platform Raspbian:

   ```
   sudo reboot
   ```

   Untuk OpenWrt platform:

   ```
   reboot
   ```

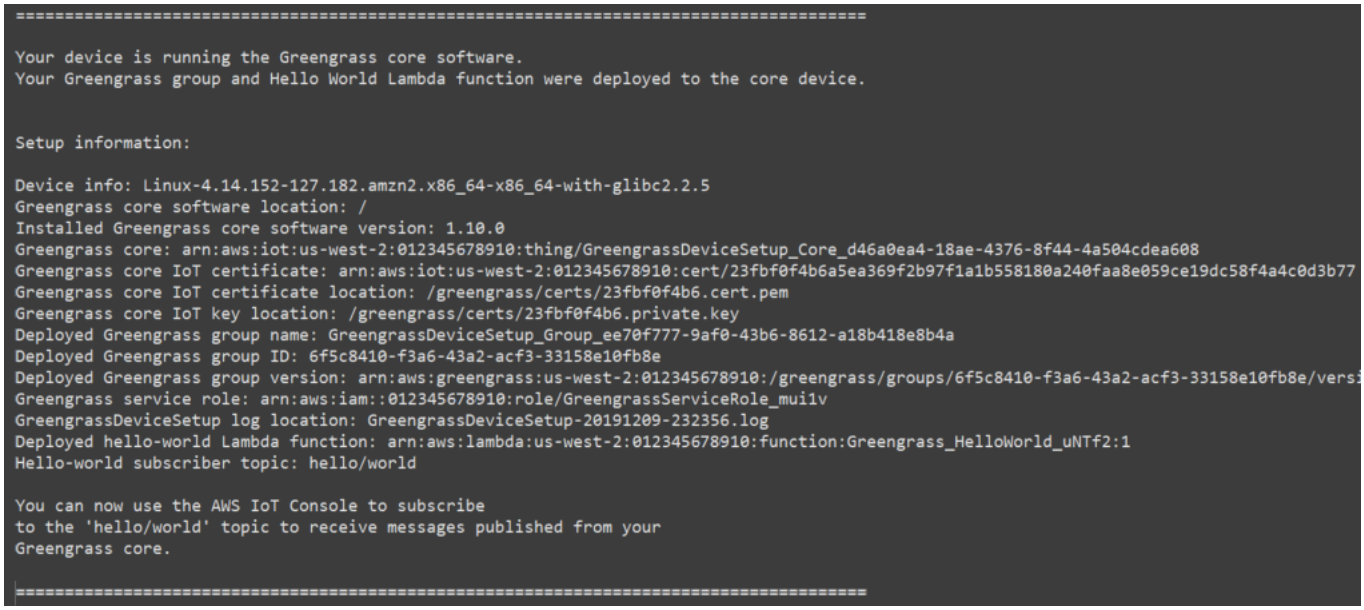
b. Restart skrip. Anda harus menyertakan kredensial Anda dalam perintah, tetapi tidak nilai-nilai input lainnya. Misalnya:

   ```
   sudo -E ./gg-device-setup-latest.sh bootstrap-greengrass
   --aws-access-key-id AKIAIOSFODNN7EXAMPLE
   --aws-secret-access-key wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   --aws-session-token AQoDYXdzEJr1K...o5OytwEXAMPLE=
   ```

c. Ketika diminta apakah akan menggunakan nilai input dari sesi sebelumnya atau memulai penginstalan baru, masukkan `yes` untuk menggunakan kembali nilai input Anda.

   > **Note**
   >
   > Pada platform yang memerlukan reboot, nilai input Anda dari sesi sebelumnya, tidak termasuk kredensial, untuk sementara disimpan di `GreengrassDeviceSetup.config.info` file.

Setelah penyiapan selesai, terminal akan menampilkan pesan status sukses yang mirip dengan berikut ini.



3. Jika Anda menyertakan fungsi Lambda, penyiapan perangkat Greengrass men-deploy grup Greengrass ke perangkat core Anda. Untuk menguji fungsi Lambda, atau untuk informasi tentang cara menghapus fungsi Lambda dari grup, lanjutkan ke the section called “Verifikasi fungsi Lambda berjalan pada perangkat core” di Modul 3-1 dari tutorial Memulai.

   > **Note**
   >
   > Pastikan Wilayah AWS yang dipilih di konsol sama dengan yang Anda gunakan untuk mengonfigurasi lingkungan Greengrass Anda. Secara default, Wilayah adalah US West (Oregon).

   Jika Anda tidak menyertakan fungsi Lambda Hello World, Anda dapat membuat fungsi Lambda Anda sendiri atau mencoba fitur Greengrass lainnya. Sebagai contoh, Anda dapat menambahkan konektor Deployment aplikasi Docker ke grup Anda dan menggunakannya untuk men-deploy kontainer Docker ke perangkat core Anda.

Modul 1: Penyiapan lingkungan untuk Greengrass

Modul ini menunjukkan cara untuk mendapatkan out-of-the-box Raspberry Pi, Amazon EC2 contoh, atau perangkat lain siap untuk digunakanAWS IoT GreengrasssebagaiAWS IoT Greengrassperangkat inti

> **Tip**
>
> Atau, untuk menggunakan skrip yang mengatur perangkat core untuk Anda, lihat the section called “Quick start: penyiapan perangkat Greengrass”.

Modul ini akan memakan waktu kurang dari 30 menit untuk menyelesaikannya.

Sebelum Anda memulai, baca persyaratan untuk tutorial ini. Kemudian, ikuti petunjuk penyiapan di salah satu topik berikut. Pilih hanya topik yang berlaku untuk jenis perangkat core Anda.

Topik

- Mengatur Raspberry Pi
- Mengatur instans Amazon EC2
- Mengatur perangkat lain

> **Note**
>
> Untuk mempelajari cara menggunakan AWS IoT Greengrass berjalan dalam kontainer Docker prebuilt, lihat the section called “Jalankan AWS IoT Greengrass di kontainer Docker”.

Mengatur Raspberry Pi

Ikuti langkah-langkah dalam topik ini untuk mengatur Raspberry Pi untuk digunakan sebagaiAWS IoT Greengrassinti.

> **Tip**
>
> AWS IoT Greengrass juga menyediakan opsi lain untuk menginstal Perangkat lunak Core AWS IoT Greengrass ini. Sebagai contoh, Anda dapat menggunakan Penyiapan perangkat Greengrass untuk mengonfigurasi lingkungan Anda dan menginstal versi terbaru perangkat lunak Core AWS IoT Greengrass ini. Atau, pada platform Debian yang mendukung, Anda dapat menggunakan Manajer paket APT untuk menginstal atau meningkatkan perangkat lunak Core AWS IoT Greengrass ini. Untuk informasi selengkapnya, lihat the section called “Instal AWS IoT Greengrass perangkat lunak Core”.

Jika Anda mengatur Raspberry Pi untuk pertama kalinya, Anda harus mengikuti semua langkah ini. Jika tidak, Anda dapat melewati ke langkah 9. Namun, kami merekomendasikan bahwa Anda re-image Anda Raspberry Pi dengan sistem operasi seperti yang direkomendasikan pada langkah 2.

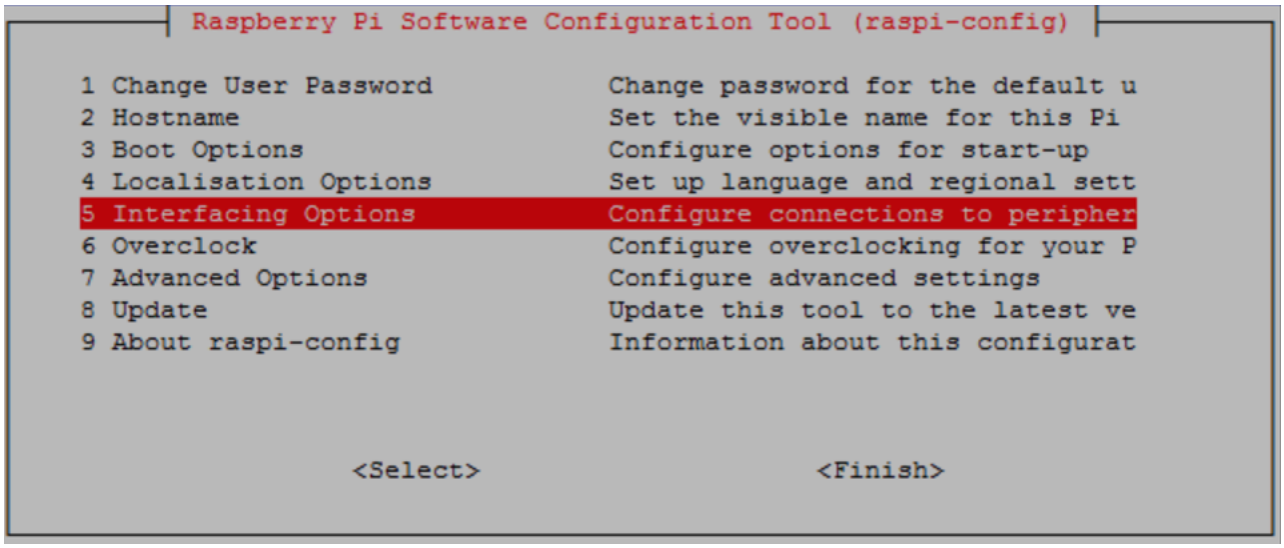
1. Unduh dan instal pemformat kartu SD seperti SD Memory Card Formatter. Masukkan kartu SD ke komputer Anda. Mulai program dan pilih drive di mana Anda telah memasukkan kartu SD Anda. Anda bisa melakukan format cepat kartu SD.
2. Unduh RaspbianBuster sistem operasi sebagai `zip` file.
3. Menggunakan alat penulisan kartu SD (seperti Etcher), ikuti petunjuk alat untuk mem-flash unduhan file `zip` ke kartu SD. Karena citra sistem operasi berukuran besar, langkah ini mungkin memerlukan waktu. Keluarkan kartu SD Anda dari komputer Anda, dan masukkan kartu microSD ke Raspberry Pi Anda.
4. Untuk boot pertama, kami sarankan Anda menghubungkan Raspberry Pi ke monitor (melalui HDMI), keyboard, dan mouse. Selanjutnya, hubungkan Pi Anda ke sumber daya USB mikro dan sistem operasi Raspbian harus dimulai.
5. Anda mungkin ingin mengonfigurasi tata letak keyboard Pi sebelum melanjutkan. Untuk melakukannya, pilih Raspberry ikon di kanan atas, pilih Preferensi lalu pilih Pengaturan Mouse dan Keyboard. Selanjutnya, pada tab Keyboard ini, pilih Keyboard Layout, lalu pilih varian keyboard yang sesuai.
6. Selanjutnya, hubungkan Raspberry Pi Anda ke internet melalui jaringan Wi-Fi atau kabel Ethernet.

> **Note**
>
> Hubungkan Raspberry Pi ke jaingan yang sama yang komputer Anda terhubung ke, dan pastikan bahwa komputer Anda dan Raspberry Pi memiliki akses internet sebelum Anda melanjutkan. Jika Anda berada di lingkungan kerja atau di belakang firewall, Anda mungkin perlu menyambungkan Pi dan komputer ke jaringan tamu untuk mendapatkan kedua perangkat di jaringan yang sama. Namun, pendekatan ini mungkin memutuskan komputer Anda dari sumber daya jaringan lokal, seperti intranet Anda. Salah satu solusinya adalah menghubungkan Pi ke jaringan Wi-Fi tamu dan menghubungkan komputer Anda ke jaringan Wi-Fi tamu dan jaringan lokal Anda melalui kabel Ethernet. Dalam konfigurasi ini, komputer Anda harus dapat terhubung ke Raspberry Pi melalui jaringan Wi-Fi tamu dan sumber daya jaringan lokal Anda melalui kabel Ethernet.

7. Anda harus mengatur SSH di Pi Anda untuk terhubung dari jarak jauh ke sana. Pada Anda Raspberry Pi, buka Jendela terminal dan jalankan perintah berikut:

```
sudo raspi-config
```

Anda akan melihat berikut ini:



Gulung ke bawah dan pilih Opsi Antarmuka lalu pilih P2 SSH. Ktika diminta, pilih Ya. (Gunakan Tab kunci yang diikuti oleh Enter). SSH sekarang harus diaktifkan. Pilih OKE. Gunakan Tab kunci untuk memilih Selesai lalu tekan Enter. Jika Raspberry Pi tidak melakukan reboot secara otomatis, jalankan perintah berikut:

```
sudo reboot
```

8. Pada Raspberry Pi, jalankan perintah berikut di terminal:

```
hostname -I
```

Ini mengembalikan alamat IP Raspberry Pi Anda.

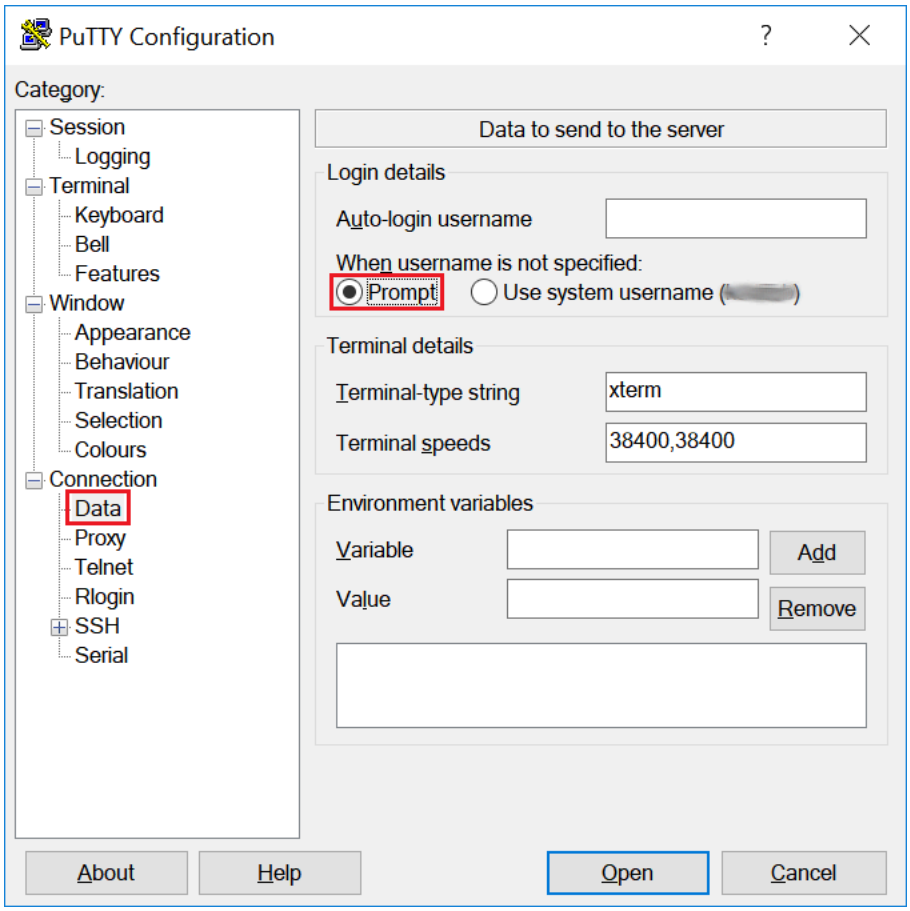
> **Note**
>
> Untuk berikut ini, jika Anda menerima pesan sidik jari kunci ECDSA (`Are you sure you want to continue connecting (yes/no)?`), masukkan `yes`. Kata sandi default untuk Raspberry Pi adalah **`raspberry`**.

Jika Anda menggunakan macOS, buka jendela terminal dan masukkan yang berikut:

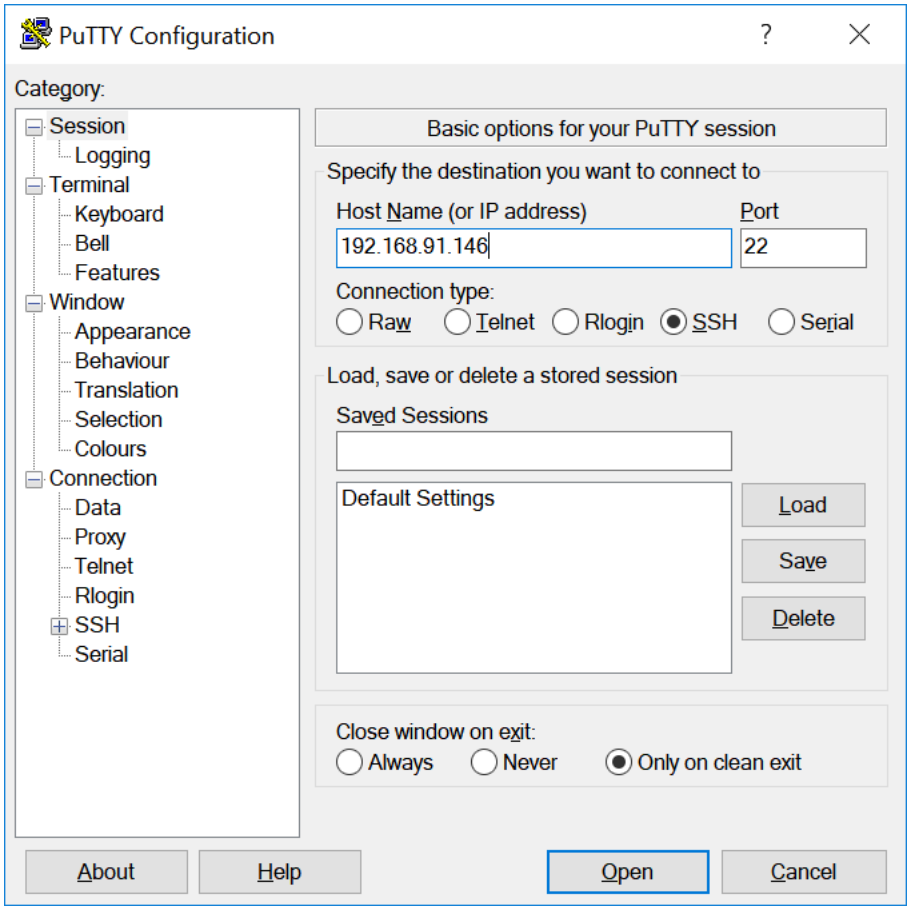
```
ssh pi@IP-address
```

Alamat IP adalah alamat IP Anda Raspberry Pi yang Anda diperoleh dengan menggunakan perintah `hostname -I` ini.

Jika Anda menggunakan Windows, Anda perlu menginstal dan mengonfigurasi PuTTY. Perluas Koneksi, pilih Data, dan pastikan bahwa Prompt dipilih:

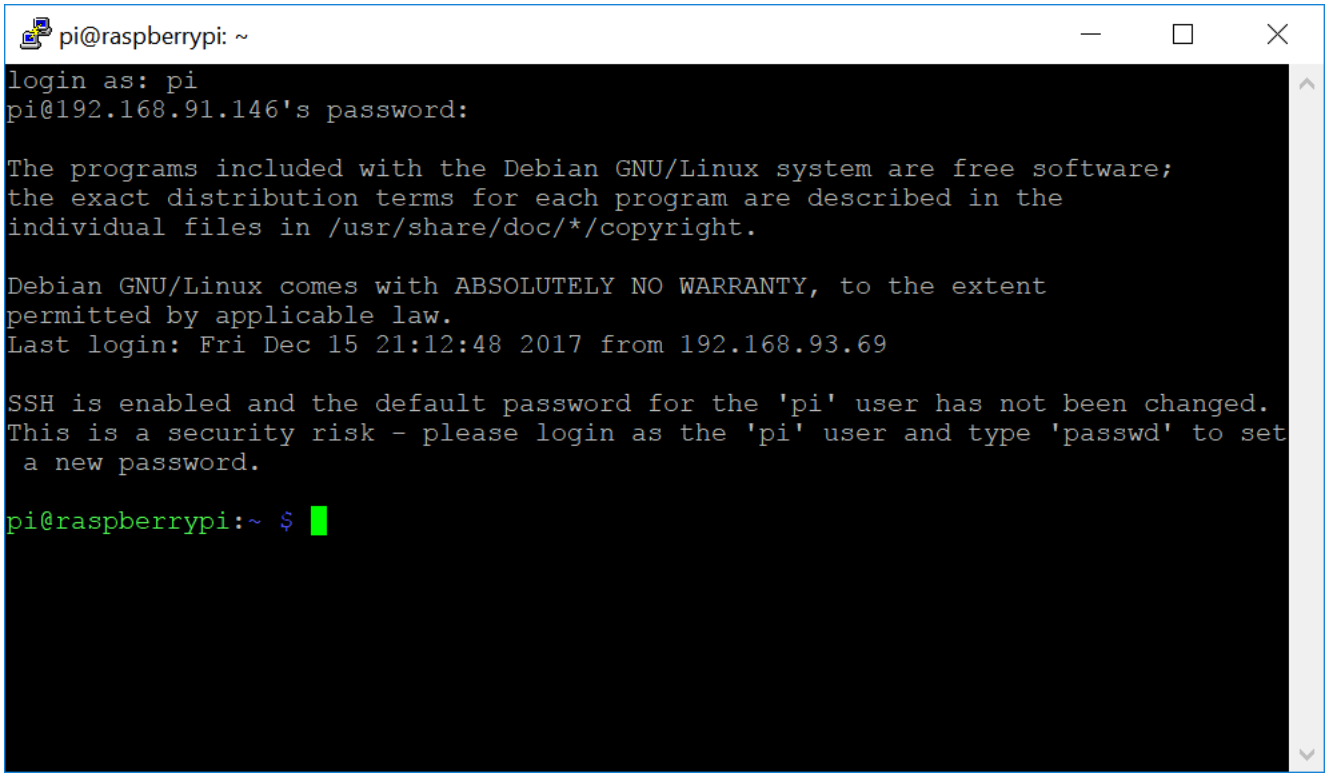


Selanjutnya, pilih Sesi, masukkan alamat IP Raspberry Pi, lalu pilih Buka menggunakan pengaturan default.



Jika peringatan keamanan PuTTY ditampilkan, pilih Ya.

Default Raspberry Pi login dan kata sandi adalah **`pi`** dan **`raspberry`**, masing-masing.



> **Note**
>
> Jika komputer Anda terhubung ke jaringan jarak jauh menggunakan VPN, Anda mungkin mengalami kesulitan menghubungkan dari komputer ke Raspberry Pi menggunakan SSH.

9. Anda sekarang siap untuk mengatur Raspberry Pi untuk AWS IoT Greengrass. Pertama, jalankan perintah berikut dari jendela terminal Raspberry Pi lokal atau jendela terminal SSH:

> **Tip**
>
> AWS IoT Greengrass juga menyediakan opsi lain untuk menginstal AWS IoT Greengrass perangkat lunak Core. Sebagai contoh, Anda dapat menggunakan Penyiapan perangkat Greengrass untuk mengonfigurasi lingkungan Anda dan menginstal versi terbaru perangkat lunak Core AWS IoT Greengrass ini. Atau, pada platform Debian yang mendukung, Anda dapat menggunakan Manajer paket APT untuk menginstal atau meningkatkan perangkat lunak Core AWS IoT Greengrass ini. Untuk informasi selengkapnya, lihat the section called “Instal AWS IoT Greengrass perangkat lunak Core”.

```
sudo adduser --system ggc_user
sudo addgroup --system ggc_group
```

10. Untuk meningkatkan keamanan pada perangkat Pi, aktifkan perlindungan hardlink dan softlink (symlink) pada sistem operasi di startup.

    a. Navigasi ke `98-rpi.conf` file.

    ```
    cd /etc/sysctl.d
    ls
    ```

    > **Note**
    >
    > Jika Anda tidak melihat `98-rpi.conf` file, ikuti petunjuk di `README.sysctl` file.

    b. Gunakan editor teks (seperti Leafpad, GNU nano, atau vi) untuk menambahkan dua baris berikut ke akhir file. Anda mungkin harus menggunakan `sudo` perintah untuk mengedit sebagai root (sebagai contoh, `sudo nano 98-rpi.conf`).

    ```
    fs.protected_hardlinks = 1
    fs.protected_symlinks = 1
    ```

    c. Reboot Pi.

    ```
    sudo reboot
    ```

    Setelah sekitar satu menit, hubungkan ke Pi menggunakan SSH lalu jalankan perintah berikut untuk mengonfirmasi perubahan:

    ```
    sudo sysctl -a 2> /dev/null | grep fs.protected
    ```

    Anda harus melihat `fs.protected_hardlinks = 1` dan `fs.protected_symlinks = 1`.

11. Edit file boot baris perintah Anda untuk mengaktifkan dan memasang cgroups memori. Ini mengizinkan AWS IoT Greengrass untuk mengatur batas memori untuk fungsi Lambda. Cgroups juga diperlukan untuk menjalankan AWS IoT Greengrass dalam mode kontainerisasi default.

a. Arahkan ke direktori `boot` Anda.

```
cd /boot/
```

b. Gunakan editor teks untuk membuka `cmdline.txt`. Tambahkan berikut ke akhir baris yang ada, bukan sebagai baris baru. Anda mungkin harus menggunakan `sudo` perintah untuk mengedit sebagai root (sebagai contoh, `sudo nano cmdline.txt`).

```
cgroup_enable=memory cgroup_memory=1
```

c. Sekarang reboot Pi.

```
sudo reboot
```

Raspberry Pi Anda sekarang harus siap untuk AWS IoT Greengrass.

12. Tidak wajib. Instal waktu aktif Java 8, yang diperlukan oleh pengelola pengaliran. Tutorial ini tidak menggunakan pengelola pengaliran, tetapi menggunakan alur kerja Pembuatan Grup Default yang mengaktifkan pengelola pengaliran secara default. Gunakan perintah berikut untuk menginstal waktu aktif Java 8 pada perangkat core, atau menonaktifkan pengelola pengaliran sebelum Anda men-deploy grup Anda. Petunjuk untuk menonaktifkan pengelola pengaliran disediakan dalam Modul 3.

```
sudo apt install openjdk-8-jdk
```

13. Untuk memastikan bahwa Anda memiliki semua dependensi yang diperlukan, unduh dan jalankan pemeriksa dependensi Greengrass dariAWS IoT GreengrassSampelrepositori pada GitHub. Perintah ini unzip dan jalankan skrip pemeriksa depedensi di `Downloads` direktori.

> **Note**
>
> Pemeriksa ketergantungan mungkin gagal jika Anda menjalankan versi 5.4.51 Raspbian kernel. Versi ini tidak memasang memori cgroups dengan benar. Hal ini dapat menyebabkan fungsi Lambda berjalan dalam mode kontainer gagal.
> Untuk informasi lebih lanjut tentang memperbarui kernel, lihat Cgroups tidak dimuat setelah meningkatkan kernel dalam forum Raspberry Pi.

```
cd /home/pi/Downloads
mkdir greengrass-dependency-checker-GGCv1.11.x
cd greengrass-dependency-checker-GGCv1.11.x
wget https://github.com/aws-samples/aws-greengrass-samples/raw/master/greengrass-
dependency-checker-GGCv1.11.x.zip
unzip greengrass-dependency-checker-GGCv1.11.x.zip
cd greengrass-dependency-checker-GGCv1.11.x
sudo modprobe configs
sudo ./check_ggc_dependencies | more
```

Di mana `more` muncul, tekan Spacebar kunci untuk menampilkan layar teks lainnya.

> **Important**
>
> Tutorial ini membutuhkan waktu aktif Python 3.7 untuk menjalankan fungsi Lambda lokal. Ketika pengelola pengaliran diaktifkan, itu juga membutuhkan waktu aktif Java 8. Jika skrip `check_ggc_dependencies` menghasilkan peringatan tentang prasyarat waktu aktif yang hilang ini, pastikan untuk menginstalnya sebelum Anda melanjutkan. Anda dapat mengabaikan peringatan tentang prasyarat waktu aktif opsional lainnya yang hilang.

Untuk informasi tentang modprobe perintah, jalankan man modprobe di terminal.

Konfigurasi Raspberry Pi Anda selesai. Lanjutkan ke the section called “Modul 2: MenginstalAWS IoT GreengrassPerangkat lunak inti”.

Mengatur instans Amazon EC2

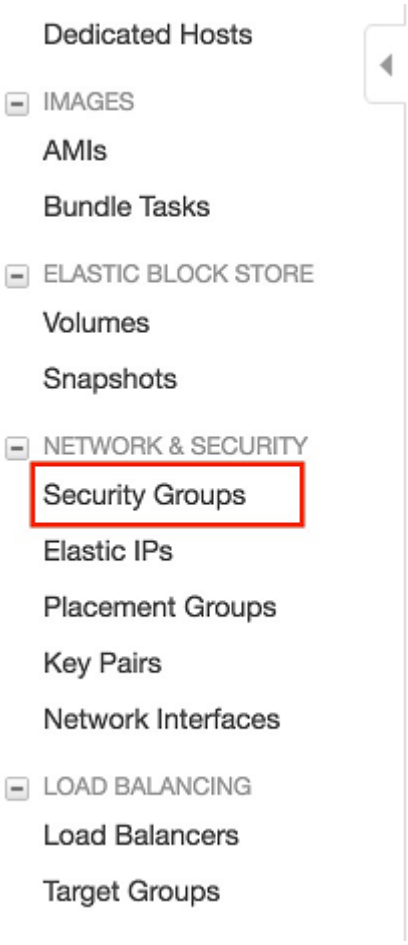
Ikuti langkah-langkah dalam topik ini untuk mengatur instans Amazon EC2 untuk digunakan sebagai AWS IoT Greengrass core.

> **Tip**
>
> Atau, untuk menggunakan skrip yang mengatur lingkungan Anda dan menginstal AWS IoT Greengrass perangkat lunak Core untuk Anda, lihat the section called “Quick start: penyiapan perangkat Greengrass”.

Meskipun Anda dapat menyelesaikan tutorial ini menggunakan instans Amazon EC2, AWS IoT Greengrass idealnya harus digunakan dengan perangkat keras fisik. Kami merekomendasikan bahwa Anda mengatur Raspberry Pi sebagai ganti menggunakan instans Amazon EC2 ketika mengizinkan. Jika Anda menggunakan Raspberry Pi, Anda tidak perlu mengikuti langkah-langkah dalam topik ini.

1. Masuk ke AWS Management Console dan meluncurkan instans Amazon EC2 menggunakan Amazon Linux AMI. Untuk informasi tentang instans Amazon EC2, lihat Panduan Memulai Amazon EC2.
2. Setelah instans Amazon EC2 Anda berjalan, aktifkan port 8883 untuk mengizinkan komunikasi MQTT masuk sehingga perangkat lain dapat terhubung dengan core AWS IoT Greengrass ini.

   a. Dalam panel navigasi konsol Amazon EC2, pilih Grup Keamanan.



b. Pilih grup keamanan untuk contoh yang Anda baru saja diluncurkan, lalu pilihAturan ke dalamtab.

c. MemiilihEdit aturan masuk.

   Untuk mengaktifkan port 8883, Anda menambahkan aturan TCP kustom untuk grup keamanan. Untuk informasi lebih lanjut, lihat Penambahan aturan ke sebuah grup keamanan dalam Panduan Pengguna Amazon EC2 untuk Instans Linux.

d. PadaEdit aturan masukhalaman, pilihTambahkan peraturan, masukkan pengaturan berikut, lalu pilihSimpan.

   - Untuk Jenis, pilih Aturan TCP Kustom.
   - Untuk Rentang port, masukkan **`8883`**.
   - Untuk Sumber, pilih Di mana saja.
   - Untuk Description (Deskripsi), masukkan **`MQTT Communications`**.

3. Connect ke instans Amazon EC2 Anda.

   a. Dalam panel navigasi, pilih Instans, pilih instans Anda, lalu pilih Connect.

   b. Ikuti petunjuk di Connect ke Instans Anda untuk terhubung ke instans Anda dengan menggunakan SSH dan file kunci privat Anda.

   Anda dapat menggunakan PuTTY untuk Windows atau Terminal untuk macOS. Untuk informasi lebih lanjut, lihat Connect ke Instans Linux Anda dalam Panduan Pengguna Amazon EC2 untuk Instans Linux.

   Anda sekarang siap untuk mengatur instans Amazon EC2 Anda untuk AWS IoT Greengrass.

4. Setelah Anda terhubung ke instans Amazon EC2, buat `ggc_user` dan `ggc_group` akun:

```
sudo adduser --system ggc_user
sudo groupadd --system ggc_group
```

> **Note**
>
> Jika perintah `adduser` tidak tersedia di sistem Anda, gunakan perintah berikut.
>
> ```
> sudo useradd --system ggc_user
> ```

5. Untuk meningkatkan keamanan, pastikan bahwa perlindungan hardlink dan softlink (symlink) diaktifkan pada sistem operasi contoh Amazon EC2 saat startup.

> **Note**
>
> Langkah-langkah untuk mengaktifkan perlindungan hardlink dan softlink bervariasi menurut sistem operasi. Lihat dokumentasi untuk distribusi Anda.

   a. Jalankan perintah berikut ini untuk memeriksa apakah perlindungan hardlink dan softlink diaktifkan:

```
sudo sysctl -a | grep fs.protected
```

Jika hardlink dan softlink diatur ke 1, perlindungan Anda diaktifkan dengan benar. Lanjutkan ke langkah 6.

> **Note**
>
> Softlink diwakili oleh `fs.protected_symlinks`.

b. Jika hardlink dan softlink tidak diatur ke 1, aktifkan perlindungan ini. Arahkan ke file konfigurasi sistem Anda.

```
cd /etc/sysctl.d
ls
```

c. Menggunakan editor teks favorit Anda (Leafpad, GNU nano, atau vi), tambahkan dua baris berikut ke akhir file konfigurasi sistem. Di Amazon Linux 1, ini adalah `00-defaults.conf` file. Di Amazon Linux 2, ini adalah `99-amazon.conf` file. Anda mungkin harus mengubah izin (menggunakan `chmod` perintah) untuk menulis ke file, atau menggunakan `sudo` perintah untuk mengedit sebagai root (sebagai contoh, `sudo nano 00-defaults.conf`).

```
fs.protected_hardlinks = 1
fs.protected_symlinks = 1
```

d. Reboot instans Amazon EC2.

```
sudo reboot
```

Setelah beberapa menit, terhubung ke contoh Anda menggunakan SSH lalu jalankan perintah berikut untuk mengonfirmasi perubahan.

```
sudo sysctl -a | grep fs.protected
```

Anda akan melihat bahwa hardlink dan softlink diatur ke 1.

6. Ekstrak dan jalankan skrip berikut untuk memasang grup kontrol Linux (cgroups). Ini mengizinkan AWS IoT Greengrass untuk mengatur batas memori untuk fungsi Lambda. Cgroups juga diperlukan untuk menjalankan AWS IoT Greengrass dalam mode kontainerisasi default.

```
curl https://raw.githubusercontent.com/tianon/cgroupfs-
mount/951c38ee8d802330454bdede20d85ec1c0f8d312/cgroupfs-mount > cgroupfs-mount.sh
```

```
chmod +x cgroupfs-mount.sh
sudo bash ./cgroupfs-mount.sh
```

Instans Amazon EC2 Anda sekarang harus siap untuk AWS IoT Greengrass.

7. Tidak wajib. Instal waktu aktif Java 8, yang diperlukan oleh pengelola pengaliran. Tutorial ini tidak menggunakan pengelola pengaliran, tetapi menggunakan alur kerja Pembuatan Grup Default yang mengaktifkan pengelola pengaliran secara default. Gunakan perintah berikut untuk menginstal waktu aktif Java 8 pada perangkat core, atau menonaktifkan pengelola pengaliran sebelum Anda men-deploy grup Anda. Petunjuk untuk menonaktifkan pengelola pengaliran disediakan dalam Modul 3.

   - Untuk distribusi berbasis Debian:

     ```
     sudo apt install openjdk-8-jdk
     ```

   - Untuk distribusi berbasis Red Hat:

     ```
     sudo yum install java-1.8.0-openjdk
     ```

8. Untuk memastikan bahwa Anda memiliki semua dependensi yang diperlukan, unduh dan jalankan pemeriksa deperdensi Greengrass dariAWS IoT GreengrassSampelrepositori pada GitHub. Perintah ini mengunduh, unzip, dan menjalankan skrip pemeriksa dependensi dalam contoh Amazon EC2 Anda.

   ```
   mkdir greengrass-dependency-checker-GGCv1.11.x
   cd greengrass-dependency-checker-GGCv1.11.x
   wget https://github.com/aws-samples/aws-greengrass-samples/raw/master/greengrass-
   dependency-checker-GGCv1.11.x.zip
   unzip greengrass-dependency-checker-GGCv1.11.x.zip
   cd greengrass-dependency-checker-GGCv1.11.x
   sudo ./check_ggc_dependencies | more
   ```

   Important

   Tutorial ini membutuhkan waktu aktif Python 3.7 untuk menjalankan fungsi Lambda lokal. Ketika pengelola pengaliran diaktifkan, itu juga membutuhkan waktu aktif Java 8. Jika skrip `check_ggc_dependencies` menghasilkan peringatan tentang prasyarat waktu aktif yang hilang ini, pastikan untuk menginstalnya sebelum Anda melanjutkan. Anda

dapat mengabaikan peringatan tentang prasyarat waktu aktif opsional lainnya yang hilang.

Konfigurasi instans Amazon EC2 Anda selesai. Lanjutkan ke the section called “Modul 2: MenginstalAWS IoT GreengrassPerangkat lunak inti”.

Mengatur perangkat lain

Ikuti langkah-langkah dalam topik ini untuk mengatur perangkat (selain Raspberry Pi) untuk digunakan sebagai core AWS IoT Greengrass Anda.

> **Tip**
>
> Atau, untuk menggunakan skrip yang mengatur lingkungan Anda dan menginstal AWS IoT Greengrass perangkat lunak Core untuk Anda, lihat the section called “Quick start: penyiapan perangkat Greengrass”.

Jika Anda baru AWS IoT Greengrass, kami merekomendasikan Anda menggunakan Raspberry Pi atau instans Amazon EC2 sebagai perangkat core Anda, dan ikuti langkah penyiapan sesuai untuk perangkat Anda.

Jika Anda berencana untuk membangun sistem berbasis Linux kustom menggunakan Proyek Yocto, Anda dapat menggunakan AWS IoT Greengrass Bitbake Recipe dari proyek `meta-aws` ini. Resep ini juga membantu Anda mengembangkan platform perangkat lunak yang mendukung AWS perangkat lunak edge untuk aplikasi tertanam. Bitbake membangun menginstal, mengonfigurasi, dan secara otomatis menjalankan AWS IoT Greengrass perangkat lunak Core di perangkat Anda.

Proyek Yocto

: Proyek kolaborasi sumber terbuka yang membantu Anda membangun sistem berbasis Linux kustom untuk aplikasi tertanam terlepas arsitektur perangkat keras. Untuk informasi lebih lanjut, lihat Proyek Yocto.

`meta-aws`

: Sebuah AWS mengelola proyek yang menyediakan Yocto recipes. Anda dapat menggunakan recipes untuk mengembangkan AWS perangkat lunak edge dalam sistem berbasis Linux yang

dibangun dengan OpenEmbedded dan Proyek Yocto. Untuk informasi lebih lanjut tentang kemampuan yang didukung oleh komunitas ini, lihat `meta-aws`proyek di GitHub.

`meta-aws-demos`

Sebuah AWS mengelola proyek yang berisi demonstrasi untuk `meta-aws` proyek. Untuk contoh lebih lanjut tentang proses integrasi, lihat `meta-aws-demos` proyek di GitHub.

Untuk menggunakan perangkat lain atau Platform yang didukung, ikuti langkah-langkah dalam topik ini.

1. Jika perangkat core Anda adalah perangkat NVIDIA Jetson, Anda harus terlebih dahulu menyalakan firmware dengan JetPack 4.3 penginstal. Jika Anda mengonfigurasi perangkat yang berbeda, lewati ke langkah 2.

   > **Note**
   >
   > Parameter JetPack versi penginstal yang Anda gunakan didasarkan pada target versi CUDA Toolkit. Petunjuk berikut menggunakan JetPack 4.3 dan CUDA Toolkit 10.0. Untuk informasi tentang cara menggunakan versi yang sesuai untuk perangkat Anda, lihat Cara Menginstal Jetpack dalam dokumentasi NVIDIA.

   a. Pada desktop fisik yang menjalankan Ubuntu 16.04 atau yang lebih baru, flash firmware dengan JetPack 4.3 installer, seperti yang dijelaskan dalamMengunduh dan Menginstal JetPack(4.3) dalam dokumentasi NVIDIA.

      Ikuti petunjuk di penginstal untuk menginstal semua paket dan dependensi di forum Jetson, yang harus terhubung ke desktop dengan kabel Micro-B.

   b. Reboot forum Anda dalam mode normal, dan hubungkan tampilan ke forum.

      > **Note**
      >
      > Saat Anda menggunakan SSH untuk terhubung ke forum Jetson, gunakan nama pengguna default (**`nvidia`**) dan kata sandi default (**`nvidia`**).

2. Jalankan perintah berikut untuk membuat pengguna `ggc_user` dan grup `ggc_group`. Perintah yang Anda jalankan berbeda, tergantung pada distribusi yang diinstal pada perangkat core Anda.

- Jika perangkat core Anda menjalankan OpenWRT, jalankan perintah berikut:

```
opkg install shadow-useradd
opkg install shadow-groupadd
useradd --system ggc_user
groupadd --system ggc_group
```

- Jika tidak, jalankan perintah berikut:

```
sudo adduser --system ggc_user
sudo addgroup --system ggc_group
```

> **Note**
>
> Jika perintah `addgroup` tidak tersedia di sistem Anda, gunakan perintah berikut.
>
> ```
> sudo groupadd --system ggc_group
> ```

3. Tidak wajib. Instal waktu aktif Java 8, yang diperlukan oleh pengelola pengaliran. Tutorial ini tidak menggunakan pengelola pengaliran, tetapi menggunakan alur kerja Pembuatan Grup Default yang mengaktifkan pengelola pengaliran secara default. Gunakan perintah berikut untuk menginstal waktu aktif Java 8 pada perangkat core, atau menonaktifkan pengelola pengaliran sebelum Anda men-deploy grup Anda. Petunjuk untuk menonaktifkan pengelola pengaliran disediakan dalam Modul 3.

   - Untuk distribusi berbasis Debian atau berbasis Ubuntu:

   ```
   sudo apt install openjdk-8-jdk
   ```

   - Untuk distribusi berbasis Red Hat:

   ```
   sudo yum install java-1.8.0-openjdk
   ```

4. Untuk memastikan bahwa Anda memiliki semua dependensi yang diperlukan, unduh dan jalankan pemeriksa depedensi Greengrass dari repositori AWS IoT Greengrass sampel di GitHub. Memerintahkan unzip dan menjalankan skrip pemeriksa dependensi.

```
mkdir greengrass-dependency-checker-GGCv1.11.x
```

```
cd greengrass-dependency-checker-GGCv1.11.x
wget https://github.com/aws-samples/aws-greengrass-samples/raw/master/greengrass-
dependency-checker-GGCv1.11.x.zip
unzip greengrass-dependency-checker-GGCv1.11.x.zip
cd greengrass-dependency-checker-GGCv1.11.x
sudo ./check_ggc_dependencies | more
```

> **Note**
>
> Skrip `check_ggc_dependencies` berjalan pada AWS IoT Greengrass platform yang didukung dan memerlukan perintah sistem Linux tertentu. Untuk informasi lebih lanjut, lihat pemeriksa dependensi Readme.

5. Instal semua dependensi yang diperlukan pada perangkat Anda, seperti yang ditunjukkan oleh output pemeriksa dependensi. Untuk dependensi tingkat kernel yang hilang, Anda mungkin harus mengkompilasi ulang kernel Anda. Untuk memasang grup kontrol Linux (`cgroups`), Anda dapat menjalankan skrip cgroupfs-mount ini. Ini mengizinkan AWS IoT Greengrass untuk mengatur batas memori untuk fungsi Lambda. Cgroups juga diperlukan untuk menjalankan AWS IoT Greengrass dalam mode kontainerisasi default.

   Jika tidak ada kesalahan muncul dalam output, AWS IoT Greengrass harus dapat berjalan dengan sukses di perangkat Anda.

   > **Important**
   >
   > Tutorial ini membutuhkan waktu aktif Python 3.7 untuk menjalankan fungsi Lambda lokal. Ketika pengelola pengaliran diaktifkan, itu juga membutuhkan waktu aktif Java 8. Jika skrip `check_ggc_dependencies` menghasilkan peringatan tentang prasyarat waktu aktif yang hilang ini, pastikan untuk menginstalnya sebelum Anda melanjutkan. Anda dapat mengabaikan peringatan tentang prasyarat waktu aktif opsional lainnya yang hilang.

   Untuk daftar AWS IoT Greengrass persyaratan dan dependensi, lihat the section called “Platform dan persyaratan yang didukung”.

Modul 2: MenginstalAWS IoT GreengrassPerangkat lunak inti

Modul ini menunjukkan cara menginstal AWS IoT Greengrass perangkat lunak Core pada perangkat yang Anda pilih. Dalam modul ini, Anda pertama kali membuat Greengrass dan core. Kemudian, Anda mengunduh, mengonfigurasi, dan mulai perangkat lunak pada perangkat core Anda. Untuk informasi lebih lanjut tentang AWS IoT Greengrass fungsionalitas perangkat lunak Core, lihat the section called “Mengonfigurasi AWS IoT Greengrass core”.

Sebelum Anda mulai, pastikan bahwa Anda telah menyelesaikan langkah-langkah penyiapan dalam Modul 1 untuk perangkat yang Anda pilih.

> **Tip**
>
> AWS IoT Greengrass juga menyediakan opsi lain untuk menginstal AWS IoT Greengrass perangkat Core. Sebagai contoh, Anda dapat menggunakan Penyiapan perangkat Greengrass untuk mengonfigurasi lingkungan Anda dan menginstal versi terbaru perangkat lunak Core AWS IoT Greengrass ini. Atau, pada platform Debian yang mendukung, Anda dapat menggunakan Manajer paket APT untuk menginstal atau meningkatkan perangkat lunak Core AWS IoT Greengrass ini. Untuk informasi selengkapnya, lihat the section called “Instal AWS IoT Greengrass perangkat lunak Core”.

Modul ini akan memakan waktu kurang dari 30 menit untuk menyelesaikannya.

Topik

- PenyediaanAWS IoThal untuk digunakan sebagai inti Greengrass
- BuatAWS IoT Greengrass grup untuk inti
- Instal dan jalankanAWS IoT Greengrasspada perangkat inti

PenyediaanAWS IoThal untuk digunakan sebagai inti Greengrass

Greengrassintiadalah perangkat yang menjalankanAWS IoT GreengrassPerangkat lunak inti untuk mengelola proses IoT lokal. Untuk mengatur inti Greengrass, Anda membuatAWS IoT hal, yang mewakili perangkat atau entitas logis yang terhubung keAWS IoT. Saat Anda mendaftarkan perangkat sebagaiAWS IoTHal, perangkat itu dapat menggunakan sertifikat digital dan kunci yang memungkinkan untuk mengaksesAWS IoT. Anda menggunakanAWS IoTkebijakanuntuk memungkinkan perangkat untuk berkomunikasi denganAWS IoTdanAWS IoT Greengrasslayanan.

Di bagian ini, Anda mendaftarkan perangkat Anda sebagaiAWS IoTHal untuk menggunakannya sebagai Greengrass core.

Untuk membuat objek AWS IoT

1. Navigasikan ke konsol AWS IoT tersebut.
2. Di bawahKelola, PerluasSemua perangkat, dan kemudian pilihMemiilih.
3. PadaMemiilihhalaman, pilihMembuat hal-hal.
4. PadaMembuat hal-halhalaman, pilihMembuat hal, dan kemudian pilihSelanjutnya.
5. PadaTentukan properti halhalaman, lakukan hal berikut:

   a. UntukNama Hal, masukkan nama yang mewakili perangkat Anda, seperti**`MyGreengrassV1Core`**.

   b. Pilih Selanjutnya.
6. PadaMengonfigurasi sertifikat perangkathalaman, pilihSelanjutnya.
7. PadaLampirkan kebijakan ke sertifikathalaman, lakukan salah satu hal berikut:

   - Pilih kebijakan yang ada yang memberikan izin yang diperlukan inti, lalu pilihMembuat hal-hal.

     Modal terbuka di mana Anda dapat mengunduh sertifikat dan kunci yang digunakan perangkat untuk terhubung keAWS Cloud.
   - Buat melampirkan kebijakan baru yang memberikan izin perangkat inti. Lakukan hal berikut:

     a. Pilih Buat kebijakan.

        ParameterBuat kebijakanhalaman terbuka di tab baru.

     b. PadaBuat kebijakanhalaman, lakukan hal berikut:

        i. UntukNama kebijakan, masukkan nama yang menjelaskan kebijakan, seperti**`GreengrassV1CorePolicy`**.

        ii. PadaPernyataan kebijakantab, di bawahDokumen kebijakan,JSON.

        iii. Masukkan dokumen kebijakan Kebijakan ini memungkinkan inti untuk berkomunikasi denganAWS IoT Corelayanan, berinteraksi dengan bayangan perangkat, dan berkomunikasi denganAWS IoT Greengrasslayanan. Untuk informasi tentang cara membatasi akses kebijakan ini berdasarkan kasus

penggunaan Anda, lihatMinimal AWS IoT kebijakan untuk AWS IoT Greengrass perangkat core.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Subscribe",
        "iot:Connect",
        "iot:Receive"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:GetThingShadow",
        "iot:UpdateThingShadow",
        "iot:DeleteThingShadow"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "greengrass:*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

iv. MemiilihBuatuntuk membuat kebijakan

c. Kembali ke tab browser denganLampirkan kebijakan ke sertifikathalaman terbuka. Lakukan hal berikut:

   i. DiKebijakandaftar, pilih kebijakan yang Anda buat, sepertiGreengrassV1CorePolicy.

      Jika Anda tidak melihat kebijakan, pilih tombol refresh.

   ii. MemiilihMembuat hal-hal.

      Modal terbuka di mana Anda dapat mengunduh sertifikat dan kunci yang digunakan inti untuk terhubungAWS IoT.

8. Kembali ke tab browser denganLampirkan kebijakan ke sertifikathalaman terbuka. Lakukan hal berikut:

   a. DiKebijakandaftar, pilih kebijakan yang Anda buat, sepertiGreengrassV1CorePolicy.

      Jika Anda tidak melihat kebijakan, pilih tombol refresh.

   b. MemiilihMembuat hal-hal.

      Modal terbuka di mana Anda dapat mengunduh sertifikat dan kunci yang digunakan inti untuk terhubungAWS IoT.

9. DiUnduh sertifikat dan kuncimodal, men-download sertifikat perangkat.

   > **Important**
   >
   > Sebelum Anda memilihSelesai, unduh sumber daya keamanan.

   Lakukan hal berikut:

   a. UntukSertifikat perangkat,Unduhuntuk mengunduh sertifikat perangkat

   b. UntukFile kunci publik,Unduhuntuk mengunduh kunci publik untuk sertifikat.

   c. UntukFile kunci privat,Unduhuntuk mengunduh file kunci privat untuk sertifikat.

   d. Review Autentikasi Server dalam AWS IoT Panduan Developer dan memilih root sertifikat CA yang sesuai. Kami merekomendasikan Anda menggunakan Amazon Trust Services (ATS) titik akhir dan ATS root sertifikat CA. Di bawahSertifikat CA Akar,Unduhuntuk sertifikat CA akar.

e. PilihSelesai.

Buat catatan ID sertifikat yang umum dalam nama file untuk sertifikat dan kunci perangkat. Anda membutuhkannya nanti.

BuatAWS IoT Greengrass grup untuk inti

AWS IoT Greengrassgrup lain yang berisi pengaturan dan informasi lainnya tentang komponennya, seperti perangkat klien, fungsi Lambda, dan konektor. Sebuah grup mendefinisikan konfigurasi untuk inti, termasuk cara komponennya dapat berinteraksi satu sama lain.

Di bagian ini, Anda membuat grup untuk inti Anda.

> **Tip**
>
> Untuk contoh yang menggunakanAWS IoT Greengrass API untuk membuat dan men-deploy grup, lihat repositori gg_group_setup di GitHub.

Untuk membuat grup untuk inti

1. Navigasikan ke konsol AWS IoT tersebut.
2. Di bawah Kelola, perluas perangkat Greengrass, dan pilih Grup (V1).

   > **Note**
   >
   > Jika Anda tidak melihat menu Greengrass s s s s s ini, ubah keWilayah AWS yang mendukungAWS IoT Greengrass V1. Untuk daftar Wilayah yang didukung, lihat AWS IoT Greengrass V1titik akhir dan kuota di bagian Referensi Umum AWS. Anda harus membuatAWS IoT hal untuk inti Anda di Wilayah di manaAWS IoT Greengrass V1 tersedia.

3. Pada halaman Grup Greengrass, pilih Buat grup.
4. Di grup Greengrass s s s s s s s s s, lakukan hal berikut:

   a. Untuk nama grup Greengrass, masukkan nama yang menggambarkan grup, seperti**`MyGreengrassGroup`**.

b. Untuk inti Greengrass, pilihAWS IoT hal yang Anda buat sebelumnya, seperti MyGreengrassV1Core.

   Konsol secara otomatis memilih sertifikat perangkat untuk Anda.

c. Pilih Create group (Buat grup).

Instal dan jalankanAWS IoT Greengrasspada perangkat inti

> **Note**
>
> Tutorial ini memberikan petunjuk bagi Anda untuk menjalankanAWS IoT GreengrassPerangkat lunak inti pada Raspberry Pi, tetapi Anda dapat menggunakan perangkat yang didukung.

Di bagian ini, Anda mengonfigurasi, menginstal, dan menjalankanAWS IoT GreengrassPerangkat lunak inti pada perangkat inti Anda.

Untuk menginstal dan menjalankanAWS IoT Greengrass

1. DariAWS IoT GreengrassPerangkat lunak intibagian dalam panduan ini, unduh bagian dalam panduan ini, unduhAWS IoT GreengrassPaket penginstalan perangkat lunak inti. Pilih paket yang paling sesuai dengan arsitektur CPU, distribusi, dan OS perangkat core Anda.

   - Untuk Raspberry Pi, unduh paket untuk arsitektur Armv7l dan sistem operasi Linux.
   - Untuk instans Amazon EC2, unduh paket untuk arsitektur x86_64 dan sistem operasi Linux.
   - Untuk NVIDIA Jetson TX2, unduh paket untuk arsitektur Armv8 (AArmv64) dan sistem operasi Linux.
   - Untuk Intel Atom, unduh paket untuk arsitektur x86_64 dan sistem operasi Linux.

2. Pada langkah sebelumnya, Anda mengunduh lima file ke komputer Anda:

   - `greengrass-`*`OS-architecture`*`-1.11.6.tar.gz`- File terkompresi ini berisiAWS IoT GreengrassPerangkat lunak inti yang berjalan pada perangkat inti.
   - *`certificateId`*`-certificate.pem.crt`- File sertifikat perangkat.
   - *`certificateId`*`-public.pem.key`- File kunci publik sertifikat perangkat.
   - *`certificateId`*`-private.pem.key`- File kunci privat sertifikat perangkat.

- `AmazonRootCA1.pem`— File otoritas sertifikat root (CA).

Pada langkah ini, Anda mentransfer file ini dari komputer Anda ke perangkat inti Anda. Lakukan hal berikut:

a. Jika Anda tidak tahu alamat IP perangkat inti Greengrass Anda, buka terminal pada perangkat inti dan jalankan perintah berikut.

> **Note**
>
> Perintah ini mungkin tidak mengembalikan alamat IP yang benar untuk beberapa perangkat. Konsultasikan dokumentasi untuk perangkat Anda untuk mengambil alamat IP perangkat Anda.

```
hostname -I
```

b. Transfer file ini dari komputer Anda ke perangkat inti Anda. Langkah transfer file bervariasi tergantung pada sistem operasi komputer Anda. Pilih sistem operasi Anda untuk langkah-langkah yang menunjukkan cara untuk mentransfer file ke perangkat Raspberry Pi Anda.

> **Note**
>
> Untuk Raspberry Pi, nama pengguna default adalah **`pi`** dan kata sandi default-nya adalah **`raspberry`**.
> Untuk NVIDIA Jetson TX2, nama pengguna default adalah **`nvidia`** dan kata sandi default-nya adalah **`nvidia`**.

Windows

Untuk mentransfer file terkompresi dari komputer Anda ke perangkat core Raspberry Pi, gunakan alat seperti WinSCP atau PuTTY pscp perintah. Untuk menggunakan pscp perintah, buka jendela Command Prompt pada komputer Anda dan jalankan hal berikut:

```
cd path-to-downloaded-files
pscp -pw Pi-password greengrass-OS-architecture-1.11.6.tar.gz pi@IP-
address:/home/pi
```

```
pscp -pw Pi-password certificateId-certificate.pem.crt pi@IP-address:/home/
pi
pscp -pw Pi-password certificateId-public.pem.key pi@IP-address:/home/pi
pscp -pw Pi-password certificateId-private.pem.key pi@IP-address:/home/pi
pscp -pw Pi-password AmazonRootCA1.pem pi@IP-address:/home/pi
```

> **Note**
>
> Nomor versi dalam perintah ini harus sesuai dengan versi paket perangkat lunak Core AWS IoT Greengrass Anda.

macOS

Untuk mentransfer file terkompresi dari Mac Anda ke perangkat core Raspberry Pi, buka jendela Terminal pada komputer Anda dan jalankan perintah berikut. Parameter*path-to-downloaded-files*biasanya adalah`~/Downloads`.

> **Note**
>
> Anda mungkin diminta untuk memasukkan dua kata sandi. Jika demikian, kata sandi pertama adalah untuk `sudo` perintah Mac dan yang kedua adalah kata sandi untuk Raspberry Pi.

```
cd path-to-downloaded-files
scp greengrass-OS-architecture-1.11.6.tar.gz pi@IP-address:/home/pi
scp certificateId-certificate.pem.crt pi@IP-address:/home/pi
scp certificateId-public.pem.key pi@IP-address:/home/pi
scp certificateId-private.pem.key pi@IP-address:/home/pi
scp AmazonRootCA1.pem pi@IP-address:/home/pi
```

> **Note**
>
> Nomor versi dalam perintah ini harus sesuai dengan versi paket perangkat lunak Core AWS IoT Greengrass Anda.

UNIX-like system

Untuk mentransfer file terkompresi dari komputer Anda ke perangkat core Raspberry Pi, buka jendela terminal pada komputer Anda dan jalankan perintah berikut:

```
cd path-to-downloaded-files
scp greengrass-OS-architecture-1.11.6.tar.gz pi@IP-address:/home/pi
scp certificateId-certificate.pem.crt pi@IP-address:/home/pi
scp certificateId-public.pem.key pi@IP-address:/home/pi
scp certificateId-private.pem.key pi@IP-address:/home/pi
scp AmazonRootCA1.pem pi@IP-address:/home/pi
```

> **Note**
>
> Nomor versi dalam perintah ini harus sesuai dengan versi paket perangkat lunak Core AWS IoT Greengrass Anda.

Raspberry Pi web browser

Jika Anda menggunakan peramban web Raspberry Pi untuk mengunduh file terkompresi, file harus berada di file Pi~`/Downloads`folder, seperti`/home/pi/Downloads`. Jika tidak, file yang dikompresi harus ada di Pi~folder, seperti`/home/pi`.

3. Pada perangkat inti Greengrass, buka terminal, dan arahkan ke folder yang berisiAWS IoT GreengrassSertifikat dan perangkat lunak inti. Ganti*path-to-transferred-files*dengan jalur tempat Anda mentransfer file pada perangkat inti. Sebagai contoh, pada Raspberry Pi, jalankan`cd /home/pi`.

   ```
   cd path-to-transferred-files
   ```

4. Buka kemasan kemasanAWS IoT GreengrassPerangkat lunak inti pada perangkat inti. Jalankan perintah berikut untuk membongkar arsip perangkat lunak yang Anda transfer ke perangkat inti. Perintah ini menggunakan`-C /`argumen untuk membuat`/greengrass`folder di folder root perangkat inti.

   ```
   sudo tar -xzvf greengrass-OS-architecture-1.11.6.tar.gz -C /
   ```

> **Note**
>
> Nomor versi dalam perintah ini harus sesuai dengan versi paket perangkat lunak Core AWS IoT Greengrass Anda.

5. Pindahkan sertifikat dan kunci keAWS IoT GreengrassFolder perangkat lunak inti. Jalankan perintah berikut untuk membuat folder untuk sertifikat dan memindahkan sertifikat dan kunci untuk itu. Ganti*path-to-transferred-files*dengan jalur di mana Anda mentransfer file pada perangkat inti, dan mengganti*certificateId*dengan ID sertifikat dalam nama file. Misalnya, pada Raspberry Pi, ganti*path-to-transferred-files*bersama**/home/pi**

```
sudo mv path-to-transferred-files/certificateId-certificate.pem.crt /greengrass/
certs
sudo mv path-to-transferred-files/certificateId-public.pem.key /greengrass/certs
sudo mv path-to-transferred-files/certificateId-private.pem.key /greengrass/certs
sudo mv path-to-transferred-files/AmazonRootCA1.pem /greengrass/certs
```

6. ParameterAWS IoT GreengrassPerangkat lunak inti menggunakan file konfigurasi yang menentukan parameter untuk perangkat lunak. File konfigurasi ini menentukan path file untuk file sertifikat danAWS Cloudtitik akhir untuk digunakan. Pada langkah ini, Anda membuatAWS IoT GreengrassFile konfigurasi perangkat lunak inti untuk inti Anda. Lakukan hal berikut:

   a. Dapatkan Amazon Resource Name (ARN) untuk perangkat inti AndaAWS IoThal-hal. Lakukan hal berikut:

      i. DiAWS IoTkonsol, di bawahKelola, di bawahPerangkat Greengrass, pilihGrup (V1).

      ii. PadaGrup Greengrasshalaman, pilih grup yang Anda buat sebelumnya.

      iii. Di bawahIkhtisar, pilihCore Greengrass.

      iv. Pada halaman detail inti, salinAWS IoThal ARN, dan simpan untuk digunakan diAWS IoT GreengrassFile konfigurasi core.

   b. DapatkanAWS IoTtitik akhir data perangkat untuk perangkatAkun AWSdi Wilayah saat ini. Perangkat menggunakan titik akhir ini untuk terhubung keAWSsebagaiAWS IoThal-hal. Lakukan hal berikut:

      i. DiAWS IoTkonsol, pilihPengaturan.

      ii. Di bawahTitik akhir data perangkat, salinTitik akhir, dan simpan untuk digunakan diAWS IoT GreengrassFile konfigurasi core.

c. BuatAWS IoT GreengrassFile konfigurasi perangkat lunak inti. Misalnya, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file .

```
sudo nano /greengrass/config/config.json
```

Ganti isi file dengan dokumen JSON berikut.

```
{
  "coreThing" : {
    "caPath": "AmazonRootCA1.pem",
    "certPath": "certificateId-certificate.pem.crt",
    "keyPath": "certificateId-private.pem.key",
    "thingArn": "arn:aws:iot:region:account-id:thing/MyGreengrassV1Core",
    "iotHost": "device-data-prefix-ats.iot.region.amazonaws.com",
    "ggHost": "greengrass-ats.iot.region.amazonaws.com",
    "keepAlive": 600
  },
  "runtime": {
    "cgroup": {
      "useSystemd": "yes"
    }
  },
  "managedRespawn": false,
  "crypto": {
    "caPath": "file:///greengrass/certs/AmazonRootCA1.pem",
    "principals": {
      "SecretsManager": {
        "privateKeyPath": "file:///greengrass/certs/certificateId-
private.pem.key"
      },
      "IoTCertificate": {
        "privateKeyPath": "file:///greengrass/certs/certificateId-
private.pem.key",
        "certificatePath": "file:///greengrass/certs/certificateId-
certificate.pem.crt"
      }
    }
  }
}
```

Kemudian, lakukan hal berikut:

- Jika Anda mengunduh sertifikat CA root Amazon yang berbeda dari Amazon Root CA 1, ganti setiap instance*AmazonRootCa1.pem*dengan nama file CA root Amazon.
- Ganti setiap contoh*certificateId*dengan ID sertifikat atas nama sertifikat dan file kunci.
- Ganti*arn: aws:iot:daerah:account-id:hal/MyGreengrassV1Core*dengan ARN hal inti Anda yang Anda simpan sebelumnya.
- Ganti*MyGreengrassV1core*Dengan nama hal-hal inti Anda.
- Ganti*device-data-prefix-ats.iot.region.amazonaws.com*denganAWS IoTendpoint data perangkat yang Anda simpan sebelumnya.
- Ganti*daerah*dengan AndaWilayah AWS.

Untuk informasi selengkapnya tentang opsi konfigurasi yang dapat Anda tentukan di file konfigurasi ini, lihatAWS IoT Greengrass file konfigurasi core.

7. Pastikan bahwa perangkat core Anda tersambung ke internet. Kemudian, mulaiAWS IoT Greengrasspada perangkat inti Anda.

```
cd /greengrass/ggc/core/
sudo ./greengrassd start
```

Anda akan melihat `Greengrass successfully started` pesan. Perhatikan PID.

> **Note**
>
> Untuk menyiapkan perangkat core untuk memulai AWS IoT Greengrass pada system boot, lihat the section called “Mulai Greengrass pada boot sistem”.

Anda dapat menjalankan perintah berikut untuk mengonfirmasi bahwa AWS IoT Greengrass Perangkat lunak core (Greengrass daemon) berfungsi. Ganti *Angka PID* dengan PID Anda:

```
ps aux | grep PID-number
```

Anda akan melihat entri untuk PID dengan path ke Greengrass daemon yang sedang berjalan (sebagai contoh, `/greengrass/ggc/packages/1.11.6/bin/daemon`). Jika Anda mengalami masalah memulai AWS IoT Greengrass, lihat Memecahkan masalah.

Modul 3 (bagian 1): Fungsi Lambda padaAWS IoT Greengrass

Modul ini menunjukkan cara untuk membuat dan men-deploy fungsi Lambda yang mengirimkan pesan MQTT dari AWS IoT Greengrass perangkat core Anda. Modul ini menjelaskan konfigurasi fungsi Lambda, langganan yang digunakan untuk mengizinkan pesan MQTT, dan deployments ke perangkat core.

Modul 3 (Bagian 2) mencakup perbedaan antara fungsi on-demand dan Lambda berumur panjang yang berjalan di AWS IoT Greengrass core.

Sebelum Anda memulai, pastikan bahwa Anda telah menyelesaikan Modul 1 dan Modul 2 dan memiliki AWS IoT Greengrass perangkat core yang berjalan.

> **Tip**
>
> Atau, untuk menggunakan skrip yang mengatur perangkat core untuk Anda, lihat the section called “Quick start: penyiapan perangkat Greengrass”. Skrip juga dapat membuat dan men-deploy fungsi Lambda yang digunakan dalam modul ini.

Modul ini akan memakan waktu sekitar 30 menit untuk menyelesaikannya.

Topik

- Buat dan paketkan fungsi Lambda
- Konfigurasikan fungsi Lambda untuk AWS IoT Greengrass
- Men-deploy konfigurasi cloud ke perangkat core Greengrass
- Verifikasi fungsi Lambda berjalan pada perangkat core

Buat dan paketkan fungsi Lambda

Contoh fungsi Python Lambda dalam modul ini menggunakan AWS IoT Greengrass Core SDK for Python untuk mempublikasikan pesan MQTT.

Dalam langkah ini, Anda:

- Unduh AWS IoT Greengrass Core SDK for Python ke komputer anda (bukan AWS IoT Greengrass perangkat core).

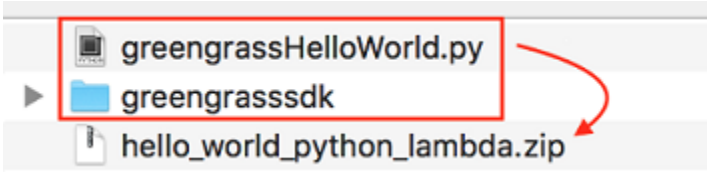
- Buat paket deployment fungsi Lambda sesuai dengan kode fungsi dan dependensinya.
- Gunakan konsol Lambda untuk membuat fungsi Lambda dan mengunggah paket deployment.
- Terbitkan versi fungsi Lambda dan buat alias yang menunjuk ke versi.

Untuk menyelesaikan modul ini, Python 3.7 harus diinstal pada perangkat core Anda.

1. Dari halaman unduh AWS IoT Greengrass SDK Core ini, unduh AWS IoT Greengrass Core SDK for Python ke komputer Anda.
2. Unzip paket download untuk mendapatkan kode fungsi Lambda dan SDK.

   Fungsi Lambda dalam modul ini menggunakan:

   - File `greengrassHelloWorld.py` di `examples\HelloWorld`. Ini kode fungsi Lambda Anda. Setiap lima detik, fungsi menerbitkan salah satu dari dua pesan yang mungkin ke `hello/world` topik.
   - Folder `greengrasssdk` ini. Ini SDK.
3. Salin folder `greengrasssdk` ke dalam folder `HelloWorld` yang berisi `greengrassHelloWorld.py`.
4. Untuk membuat paket deployment fungsi Lambda, simpan `greengrassHelloWorld.py` dan folder `greengrasssdk` ke sebuah file terkompresi `zip` bernama `hello_world_python_lambda.zip`. File py dan folder `greengrasssdk` harus berada di root direktori.

   Pada sistem seperti Unix (termasuk terminal Mac), Anda dapat menggunakan perintah berikut untuk mengemas file dan folder:

   ```
   zip -r hello_world_python_lambda.zip greengrasssdk greengrassHelloWorld.py
   ```

> **Note**
>
> Tergantung pada distribusi Anda, Anda mungkin perlu menginstal `zip` terlebih dahulu (sebagai contoh, dengan menjalankan `sudo apt-get install zip`). Perintah penginstalan untuk distribusi Anda mungkin berbeda.

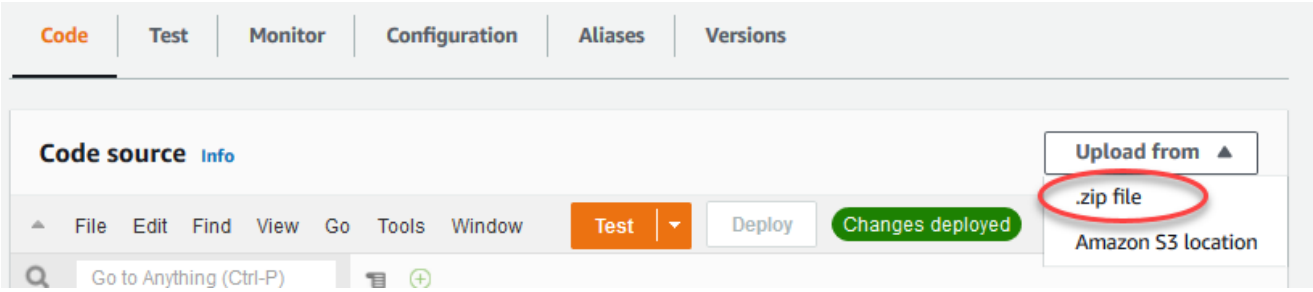
Sekarang Anda siap untuk membuat fungsi Lambda Anda dan mengunggah paket deployment.

5. Buka konsol Lambda dan pilih Buat fungsi.
6. Pilih Penulis dari scratch.
7. Beri nama fungsi Anda **`Greengrass_HelloWorld`**, dan atur bidang yang tersisa sebagai berikut:

   - Untuk Waktu aktif, pilih Python 3.7.
   - Untuk Izin, pertahankan pengaturan default. Hal ini menciptakan peran eksekusi yang memberikan izin Lambda basic. Peran ini tidak digunakan oleh AWS IoT Greengrass.

   Pilih Buat fungsi.
8. Unggah paket deployment fungsi Lambda Anda:

   a. Pada tab Kode ini, di bawah Sumber kode, pilih Unggah dari. Dari dropdown, pilih file .zip.



   b. Pilih Unggah, lalu pilih paket deployment `hello_world_python_lambda.zip` Anda. Lalu, pilih Simpan.

   c. Pada tab Kode fungsi, di bawah Pengaturan waktu aktif, pilih Edit, dan kemudian masukkan nilai-nilai berikut.

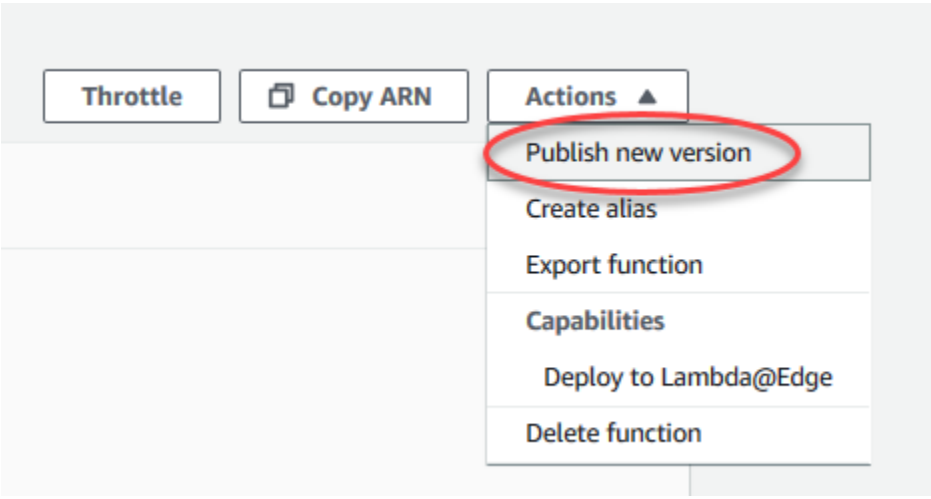
      - Untuk Waktu pengoperasian, pilih Python 3.7.
      - Untuk Handler, masukkan **`greengrassHelloWorld.function_handler`**

d. Pilih Save (Simpan).

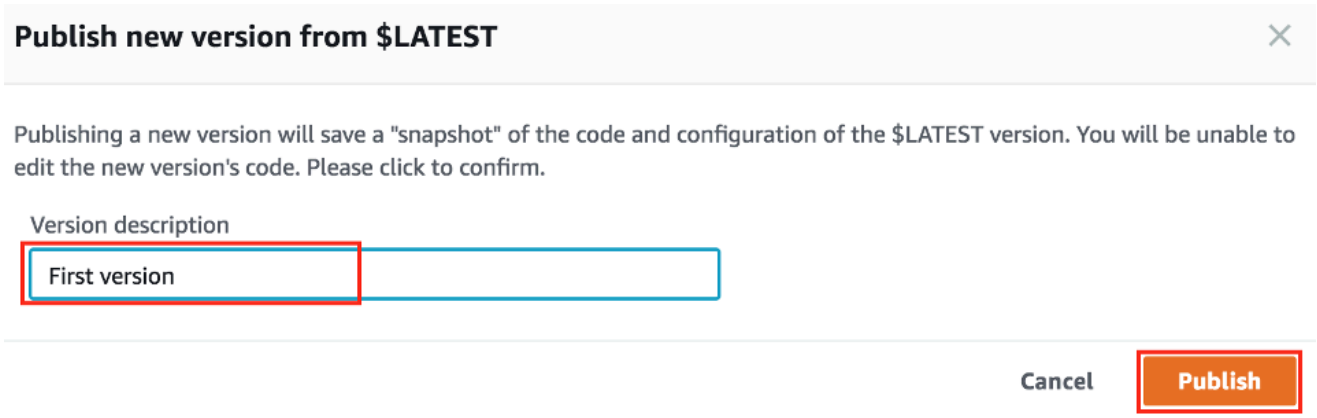
> **Note**
>
> Tombol Tes pada konsol AWS Lambda tidak bekerja dengan fungsi ini. Pada AWS IoT Greengrass Core SDK tidak berisi modul yang diperlukan untuk menjalankan fungsi Greengrass Lambda Anda secara independen di konsol AWS Lambda tersebut. Modul-modul ini (sebagai contoh, `greengrass_common`) dipasok ke fungsi setelah mereka di-deploy ke core Greengrass Anda.

9. Publikasikan fungsi Lambda:

a. Dari bagian atas halaman menu Tindakan ini, pilih Terbitkan versi baru.



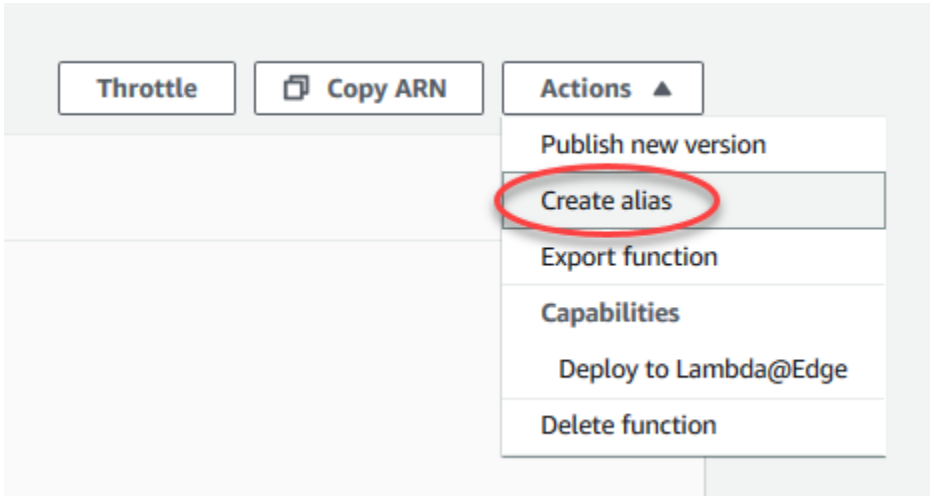
b. Untuk Deskripsi versi, masukkan **`First version`**, lalu pilih Publikasikan.



10. Buat alias untuk fungsi Lambda version:

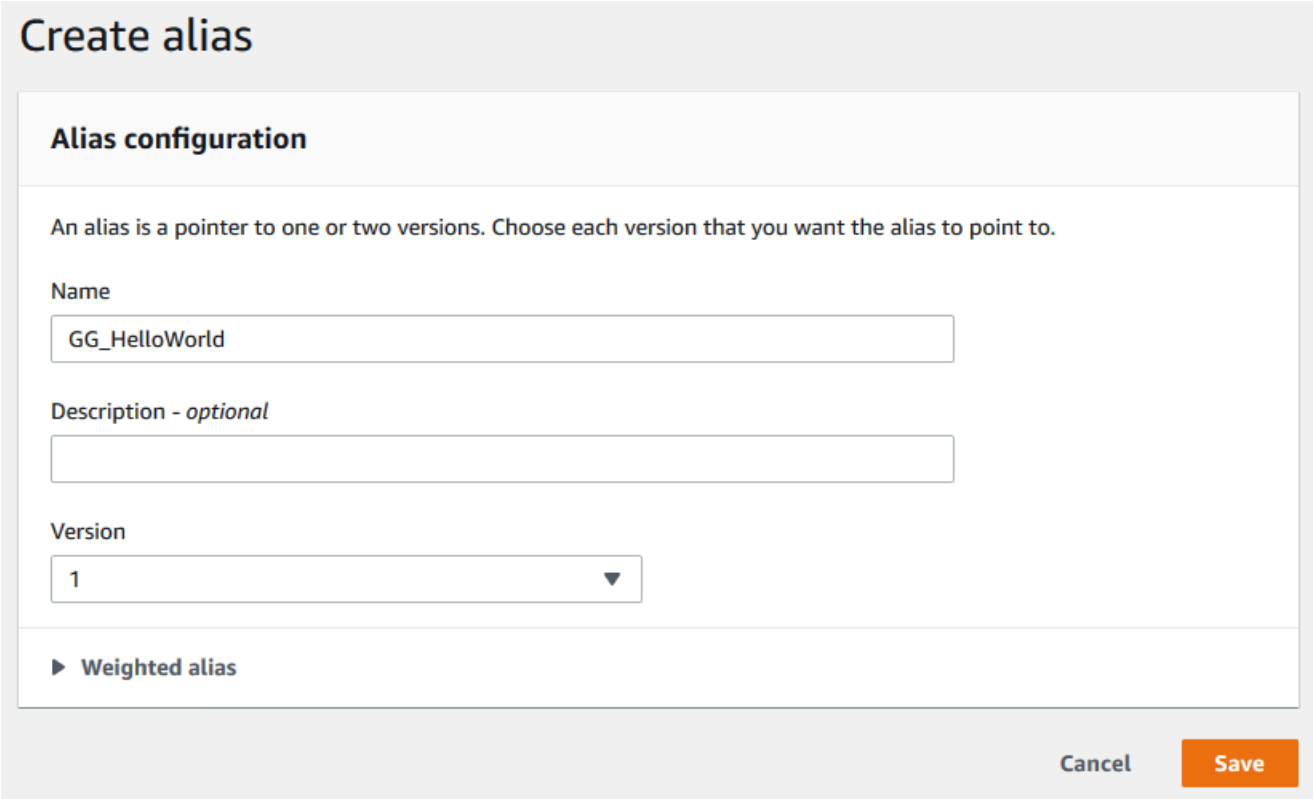
> **Note**
>
> Grup Greengrass dapat mereferensi fungsi Lambda dengan alias (direkomendasikan) atau dengan versi. Menggunakan alias membuatnya lebih mudah untuk mengelola pembaruan kode karena Anda tidak perlu mengubah tabel langganan atau definisi grup ketika kode fungsi diperbarui. Sebaliknya, Anda hanya mengarahkan alias ke versi fungsi baru.

a. Dari bagian atas halaman menu Tindakan ini, pilih Buat alias.



b. Beri nama alias **`GG_HelloWorld`**, atur versi ke **`1`** (yang sesuai dengan versi yang baru saja Anda publikasikan), lalu pilih Simpan.

> **Note**
>
> AWS IoT Greengrass tidak mendukung alias Lambda untuk $TERBARU versi.



Konfigurasikan fungsi Lambda untuk AWS IoT Greengrass

Anda sekarang siap untuk mengonfigurasi fungsi Lambda Anda untuk AWS IoT Greengrass.

Dalam langkah ini, Anda:

- Menggunakan AWS IoT konsol untuk menambahkan fungsi Lambda ke grup Greengrass Anda.
- Mengonfigurasi pengaturan grup khusus untuk fungsi Lambda.
- Menambahkan langganan ke grup yang mengizinkan fungsi Lambda untuk mempublikasikan pesan MQTT ke AWS IoT.
- Mengonfigurasi pengaturan log lokal untuk grup.

1. DiAWS IoTpanel navigasi konsol, di bawahKelola, PerluasPerangkat Greengrass, dan kemudian pilihGrup (V1).
2. Di bawah Grup Greengrass, pilih grup yang Anda buat di Modul 2.
3. Pada halaman konfigurasi grup, pilihFungsi LambdaTab, lalu gulir ke bawahFungsi Lambdabagian dan pilihTambahkan fungsi Lambda.
4. Pilih nama fungsi Lambda yang Anda buat pada langkah sebelumnya (Greengrass_HelloWorld, bukan nama alias).
5. Untuk versinya, pilihAlias: GG_HelloWorld.
6. DiKonfigurasi fungsi Lambdabagian, lakukan perubahan berikut:

   - SetPengguna dan grup sistemkepadaGunakan default grup.
   - SetKontainerisasi fungsi LambdakepadaGunakan default grup.
   - Atur Timeout ke 25 detik. Fungsi Lambda ini tidur selama 5 detik sebelum setiap pengaktifan.
   - UntukDipinned, chooseBenar.

   > **Note**
   >
   > Fungsi Lambda berumur panjang (atau disematkan) memulai secara otomatis setelah AWS IoT Greengrass dimulai dan terus berjalan dalam kontainernya sendiri. Hal ini berbeda dengan fungsi Lambda sesuai permintaan ini, yang dimulai ketika diaktifkan dan berhenti ketika tidak ada tugas yang tersisa untuk dijalankan. Untuk informasi selengkapnya, lihat the section called “Konfigurasi siklus hidup”.

7. PilihTambahkan fungsi Lambdauntuk menyimpan perubahan Anda. Untuk informasi tentang properti fungsi Lambda, lihat the section called “Mengontrol eksekusi fungsi Greengrass Lambda”.

   Selanjutnya, buat langganan yang mengizinkan fungsi Lambda untuk mengirim pesan MQTT ke AWS IoT Core.

   Sebuah fungsi Lambda Greengrass dapat bertukar pesan MQTT dengan:

   - Perangkat dalam grup Greengrass.

- Konektor dalam grup.
- Fungsi Lambda lainnya dalam grup.
- AWS IoT Core.
- Layanan bayangan lokal. Untuk informasi selengkapnya, lihat the section called “Modul 5: Berinteraksi dengan bayangan perangkat”.

Grup menggunakan langganan untuk mengontrol cara entitas ini dapat berkomunikasi satu sama lain. Langganan menyediakan interaksi yang dapat diprediksi dan lapisan keamanan.

Langganan terdiri dari sumber, target, dan topik. Sumber adalah pencetus pesan. Target adalah tujuan pesan. Topik ini mengizinkan Anda untuk memfilter data yang dikirim dari sumber ke target. Sumber atau target dapat berupa perangkat Greengrass, fungsi Lambda, konektor, bayangan perangkat, atau AWS IoT Core.

> **Note**
>
> Langganan diarahkan dalam arti bahwa pesan mengalir ke arah tertentu: dari sumber ke target. Untuk mengizinkan komunikasi dua arah, Anda harus menyiapkan dua langganan.

> **Note**
>
> Saat ini, filter topik langganan tidak mengizinkan lebih dari satu + karakter di dalam sebuah topik. Filter topik hanya mengizinkan satu karakter # di akhir topik.

Fungsi Lambda `Greengrass_HelloWorld` mengirimkan pesan hanya ke topik `hello/world` di AWS IoT Core, jadi Anda hanya perlu membuat satu langganan dari fungsi Lambda ke AWS IoT Core. Anda membuat ini di dalam langkah selanjutnya.

8. Pada halaman konfigurasi grup, pilihLangganantab, dan kemudian pilihTambahkan langganan.

   Untuk contoh yang menunjukkan Anda cara membuat langganan menggunakanAWS CLI, lihatcreate-subscription-definitiondi dalamAWS CLIReferensi Perintah.

9. DiJenis sumber, chooseFungsi Lambdadan untukSumber, chooseGreengrass_HelloWorld.
10. UntukJenis target, chooseLayanandan untukTargetmenyeleksiIoT Cloud.

11. UntukFilter topik, ENTER**`hello/world`**, dan kemudian pilihBuat langganan.
12. Konfigurasi pengaturan pencatatan grup. Untuk tutorial ini, Anda mengonfigurasi AWS IoT Greengrass komponen sistem dan fungsi Lambda yang ditentukan pengguna untuk menulis log ke sistem file perangkat inti.

    a. Pada halaman konfigurasi grup, pilihBeberapa catatanTab.

    b. DiKonfigurasi log lokalbagian, pilihEdit.

    c. PadaEdit konfigurasi catatan lokalkotak dialog, simpan nilai default untuk tingkat log dan ukuran penyimpanan, lalu pilihSimpan.

    Anda dapat menggunakan catatan untuk memecahkan masalah yang mungkin Anda alami ketika menjalankan tutorial ini. Saat memecahkan masalah, Anda dapat mengubah sementara tingkat pendataan ke Debug. Untuk informasi selengkapnya, lihat the section called “Mengakses log sistem file”.
13. Jika waktu aktif Java 8 tidak diinstal pada perangkat core Anda, Anda harus menginstalnya atau menonaktifkan pengelola pengaliran.

    > **Note**
    >
    > Tutorial ini tidak menggunakan pengelola pengaliran, tetapi menggunakan alur kerja Pembuatan Grup Default yang mengaktifkan pengelola pengaliran secara default. Jika pengelola pengaliran diaktifkan tetapi Java 8 tidak diinstal, deployment grup gagal. Untuk informasi lebih lanjut, lihat bagian persyaratan pengelola pengaliran.

    Untuk menonaktifkan pengelola pengaliran:

    a. Pada halaman pengaturan grup, pilihFungsi LambdaTab.

    b. Di bawahFungsi Lambda sistembagian, pilihManajer pengalirandan pilihlahEdit.

    c. Pilih Nonaktifkan, lalu pilih Simpan.

Men-deploy konfigurasi cloud ke perangkat core Greengrass

1. Pastikan bahwa perangkat core Greengrass Anda terhubung ke internet. Sebagai contoh, coba berhasil menavigasi ke halaman web.

2. Pastikan bahwa Greengrass daemon berjalan pada perangkat core Anda. Di terminal perangkat core Anda, jalankan perintah berikut untuk memeriksa apakah daemon sedang berjalan dan memulainya, jika diperlukan.

   a. Untuk memeriksa apakah daemon sedang berjalan:

   ```
   ps aux | grep -E 'greengrass.*daemon'
   ```

   Jika output berisi `root` entri untuk `/greengrass/ggc/packages/1.11.6/bin/daemon`, maka daemon sedang berjalan.

   b. Untuk memulai daemon:

   ```
   cd /greengrass/ggc/core/
   sudo ./greengrassd start
   ```

   Sekarang Anda siap untuk men-deploy fungsi Lambda dan konfigurasi langganan ke perangkat core Greengrass Anda.

3. DiAWS IoTpanel navigasi konsol, di bawahKelola, PerluasPerangkat Greengrass, dan kemudian pilihGrup (V1).
4. Di bawah Grup Greengrass, pilih grup yang Anda buat di Modul 2.
5. Pada halaman konfigurasi grup, pilihDeploy.
6. PadaFungsi Lambdatab, diFungsi Lambda sistembagian, pilihDetektor IP.
7. Memiiliheditdan pilihSecara otomatis mendeteksi dan mengganti titik akhir broker MQTT. Hal ini mengaktifkan perangkat untuk secara otomatis memperoleh informasi konektivitas untuk core, seperti alamat IP, DNS, dan nomor port. Deteksi otomatis direkomendasikan, namun AWS IoT Greengrass juga support titik akhir yang ditentukan secara manual. Anda hanya diminta untuk metode penemuan pertama kalinya bahwa grup di-deploy.

Deployment pertama mungkin memerlukan waktu beberapa menit. Ketika deploymen selesai, Anda akan melihat Berhasil diselesaikan di kolom Status pada halaman Deployment ini:

> **Note**
>
> Status Deployment juga ditampilkan di bawah nama grup pada header halaman.

Untuk bantuan penyelesaian masalah, lihat Memecahkan masalah.

Verifikasi fungsi Lambda berjalan pada perangkat core

1. Dari panel navigasiAWS IoTkonsol, di bawahTEST, pilihKlien uji MQTT.
2. PilihBerlangganan topiktab.
3. ENTER**`hello/world`**ke dalamFilter topikdan memperluasKonfigurasi tambahan.
4. Masukkan informasi yang tercantum di masing-masing bidang berikut:
   - Untuk Kualitas Layanan, pilih 0.
   - Untuk Tampilan muatan MQTT, pilih Tampilkan muatan sebagai string.
5. Pilih Langganan.

Dengan asumsi fungsi Lambda berjalan pada perangkat Anda, ia akan menerbitkan pesan yang mirip dengan yang berikut ke `hello/world` topik:

Meskipun fungsi Lambda terus mengirim pesan MQTT ke `hello/world` topik, jangan hentikan AWS IoT Greengrass daemon. Modul yang tersisa ditulis dengan asumsi bahwa itu berjalan.

Anda dapat menghapus fungsi dan langganan dari grup:

- Pada halaman konfigurasi grup, di bawahFungsi Lambdatab, pilih fungsi Lambda yang ingin dihapus dan pilihMenghapus.
- Pada halaman konfigurasi grup, di bawahLangganantab, pilih langganan, lalu pilihHapus.

Fungsi dan langganan dihapus dari core selama deployment grup selanjutnya.

Modul 3 (bagian 2): Lambda berfungsi padaAWS IoT Greengrass

Modul ini mengeksplorasi perbedaan antara on-demand dan fungsi Lambda berumur panjang yang berjalan di AWS IoT Greengrass Core.

Sebelum Anda memulai, jalankan perintah Pengaturan Perangkat Greengrass atau pastikan Anda telah menyelesaikan Modul 1, Modul 2, dan Modul 3 (Bagian 1).

Modul ini akan memakan waktu sekitar 30 menit untuk menyelesaikannya.

Topik

- Buat dan paket fungsi Lambda
- Mengonfigurasi fungsi Lambda berumur panjang untuk AWS IoT Greengrass
- Uji fungsi Lambda berumur panjang
- Uji fungsi Lambda sesuai permintaan

Buat dan paket fungsi Lambda

Dalam langkah ini, Anda:

- Buat paket deployment fungsi Lambda sesuai dengan kode fungsi dan dependensinya.
- Gunakan konsol Lambda untuk membuat fungsi Lambda dan mengunggah paket deployment.
- Terbitkan versi fungsi Lambda dan buat alias yang menunjuk ke versi.

1. Pada komputer Anda, pergi ke AWS IoT Greengrass Core SDK for Python yang Anda unduh dan diekstrak di the section called “Buat dan paketkan fungsi Lambda” dalam Modul 3-1.

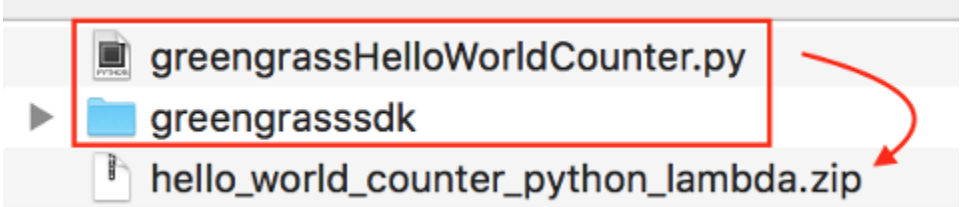
   Fungsi Lambda dalam modul ini menggunakan:

   - File `greengrassHelloWorldCounter.py` di `examples\HelloWorldCounter`. Ini kode fungsi Lambda Anda.
   - Folder `greengrasssdk` ini. Ini SDK.

2. Buat paket deployment fungsi Lambda:

   a. Salin folder `greengrasssdk` ke dalam folder `HelloWorldCounter` yang berisi `greengrassHelloWorldCounter.py`.

   b. Simpan `greengrassHelloWorldCounter.py` dan folder `greengrasssdk` untuk file `zip` bernama `hello_world_counter_python_lambda.zip`. File py dan folder `greengrasssdk` harus berada di root direktori.



   Pada sistem mirip Unix (termasuk terminal Mac) yang telah `zip` diinstal, Anda dapat menggunakan perintah berikut untuk paket file dan folder:

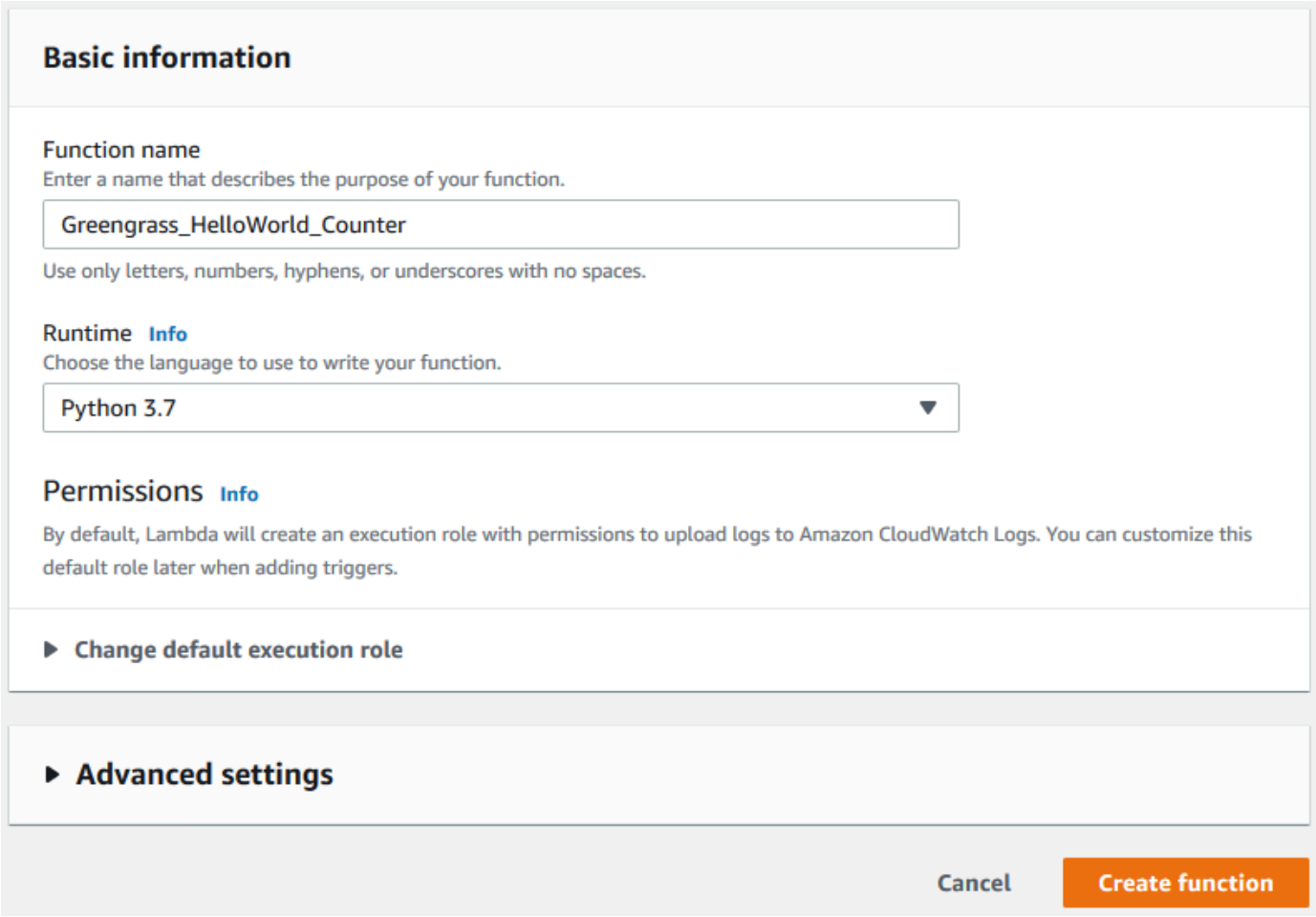
```
zip -r hello_world_counter_python_lambda.zip greengrasssdk
 greengrassHelloWorldCounter.py
```

   Sekarang Anda siap untuk membuat fungsi Lambda Anda dan mengunggah paket deployment.

3. Buka konsol Lambda dan pilih Buat fungsi.
4. Pilih Penulis dari scratch.
5. Beri nama fungsi Anda **`Greengrass_HelloWorld_Counter`**, dan atur bidang yang tersisa sebagai berikut:

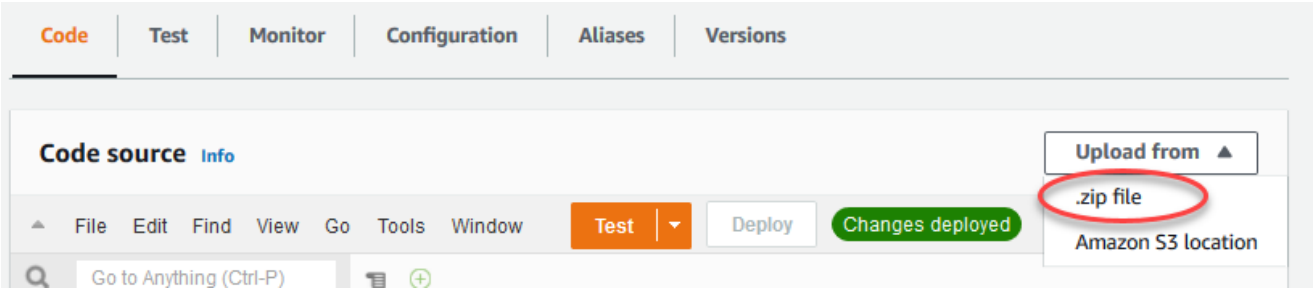
   - Untuk Waktu aktif, pilih Python 3.7.
   - Untuk Izin, pertahankan pengaturan default. Hal ini menciptakan peran eksekusi yang memberikan izin Lambda basic. Peran ini tidak digunakan oleh AWS IoT Greengrass. Atau, Anda dapat menggunakan kembali peran yang Anda buat di Modul 3-1.

   Pilih Buat fungsi.



6. Unggah paket deployment fungsi Lambda Anda.

   a. Pada tab Kode tersebut, di bawah Sumber kode, pilih unggah dari. Dari dropdown, pilih file .zip.



   b. Pilih Unggah, lalu pilih paket deployment `hello_world_counter_python_lambda.zip` Anda. Lalu, pilih Simpan.

   c. Pada tab Kode fungsi, di bawah Pengaturan waktu aktif, pilih Edit, dan kemudian masukkan nilai-nilai berikut.

      - Untuk Waktu pengoperasian, pilih Python 3.7.

- Untuk Handler, masukkan **`greengrassHelloWorldCounter.function_handler`**

d. Pilih Save (Simpan).

> **Note**
>
> Tombol Tes pada konsol AWS Lambda tidak bekerja dengan fungsi ini. Pada AWS IoT Greengrass Core SDK tidak berisi modul yang diperlukan untuk menjalankan fungsi Greengrass Lambda Anda secara independen di konsol AWS Lambda tersebut. Modul-modul ini (sebagai contoh, `greengrass_common`) dipasok ke fungsi setelah mereka di-deploy ke core Greengrass Anda.

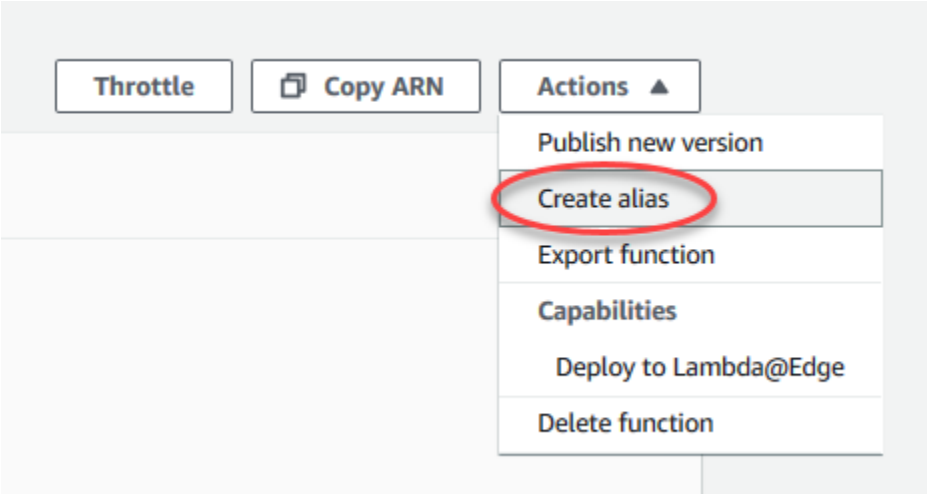
7. Terbitkan versi pertama fungsi.

   a. Dari bagian atas halaman menu Tindakan ini, pilih Terbitkan versi baru. Untuk Deskripsi Versi, masukkan **`First version`**.

   b. Pilih Terbitkan.

8. Buat alias untuk versi fungsi Lambda.

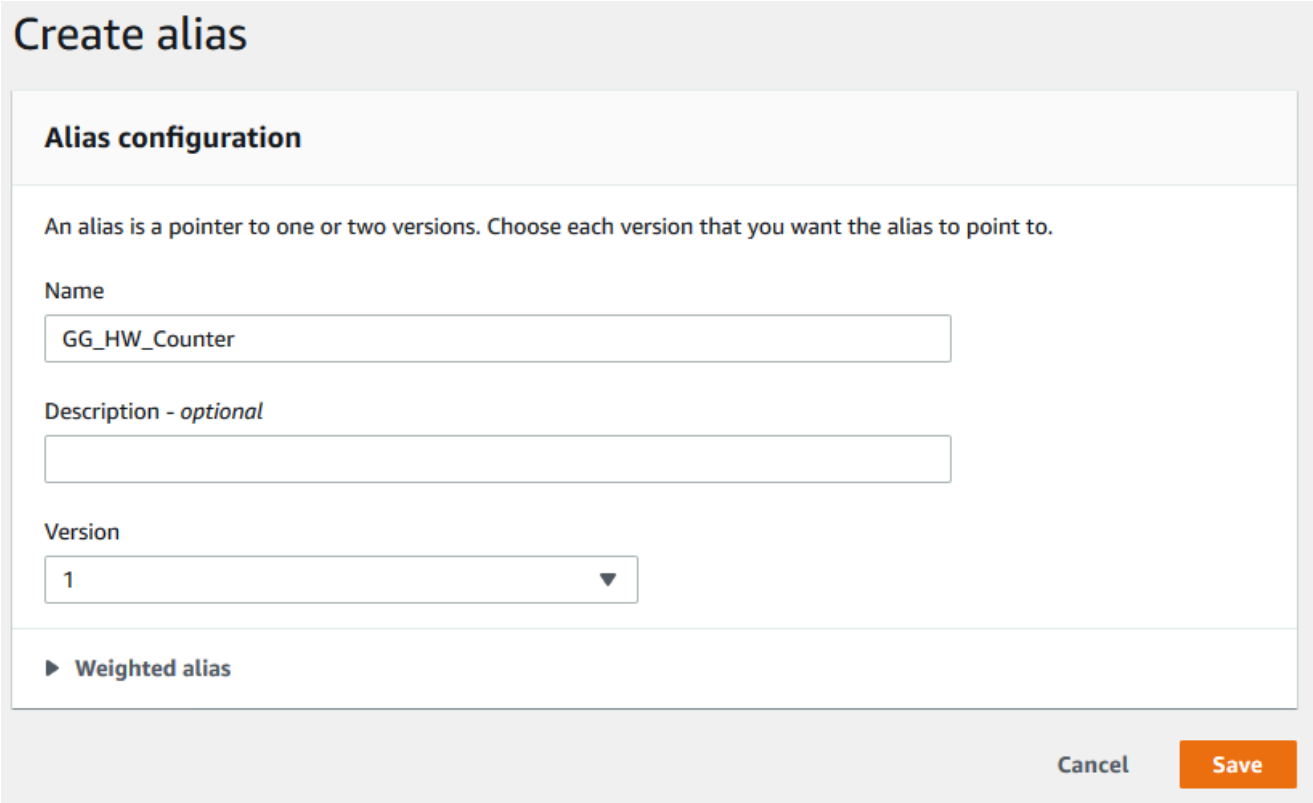
   a. Dari bagian atas halaman menu Tindakan ini, pilih Buat alias.



   b. Untuk Nama, masukkan **`GG_HW_Counter`**.

   c. Untuk Versi, pilih 1.

   d. Pilih Save (Simpan).



Alias membuat satu entitas untuk fungsi Lambda Anda yang perangkat Greengrass dapat berlangganan. Dengan cara ini, Anda tidak perlu memperbarui langganan dengan nomor versi fungsi Lambda baru setiap kali fungsi tersebut dimodifikasi.

Mengonfigurasi fungsi Lambda berumur panjang untuk AWS IoT Greengrass

Anda sekarang siap untuk mengonfigurasi fungsi Lambda Anda untuk AWS IoT Greengrass.

1. DiAWS IoTpanel navigasi konsol, di bawahKelola, perluasPerangkat Greengrass, dan kemudian pilihGrup (V1).
2. Di bawah Grup Greengrass, pilih grup yang Anda buat di Modul 2.
3. Pada halaman konfigurasi grup, pilihFungsi Lambdatab, dan kemudian di bawahFungsi Lambda saya, pilihTambahkan.
4. UntukFungsi Lambda, pilihGreengrass_HelloWorld_Penghitung.
5. UntukVersi fungsi Lambda, pilih alias ke versi yang Anda terbitkan.

6. UntukTimeout (detik), ENTER**25**. Fungsi Lambda ini tidur selama 20 detik sebelum setiap pemanggilan.
7. UntukDisematkan, pilihBenar.
8. Menjaga nilai default untuk semua bidang lainnya, dan memilihMenambahkan fungsi Lambda.

Uji fungsi Lambda berumur panjang

Sebuah Fungsi Lambda yang berumur panjang dimulai secara otomatis ketika AWS IoT Greengrass core memulai dan berjalan dalam kontainer tunggal (atau sandbox). Setiap variabel dan logika preprocessing didefinisikan di luar fungsi handler dipertahankan untuk setiap permintaan dari fungsi handler. Beberapa permintaan dari fungsi handler antri sampai permintaan sebelumnya telah dilaksanakan.

Kode `greengrassHelloWorldCounter.py` yang digunakan dalam modul ini mendefinisikan `my_counter` variabel di luar dari fungsi handler.

> **Note**
>
> Anda dapat melihat kode diAWS Lambdakonsol atauAWS IoT GreengrassCore SDK for Pythondi atas GitHub.

Dalam langkah ini, Anda membuat langganan yang mengizinkan fungsi Lambda dan AWS IoT untuk bertukar pesan MQTT. Kemudian Anda men-deploy grup dan menguji fungsi.

1. Pada halaman konfigurasi grup, pilihLangganan, dan kemudian pilihTambahkan.
2. Di bawahJenis sumber, pilihFungsi Lambda, dan kemudian pilihGreengrass_HelloWorld_Penghitung.
3. Di bawahJenis target, pilihLayanan, pilihIoT Cloud.
4. Untuk Filter topik, masukkan **`hello/world/counter`**.
5. Pilih Buat langganan.

   Langganan tunggal ini hanya berlaku dalam satu arah: dari `Greengrass_HelloWorld_Counter` Fungsi Lambda ke AWS IoT. Untuk memanggil (atau memicu) fungsi Lambda ini dari cloud, Anda harus membuat langganan dalam arah yang bertentangan.

6. Ikuti langkah 1 - 5 untuk menambahkan langganan lain yang menggunakan nilai berikut. Langganan ini mengizinkan fungsi Lambda untuk menerima pesan dari AWS IoT. Anda menggunakan langganan ini ketika Anda mengirim pesan dari AWS IoT konsol yang memanggil fungsi.

   - Untuk sumbernya, pilihLayanan, dan kemudian pilihIoT Cloud.
   - Untuk target, pilihFungsi Lambda, dan kemudian pilihGreengrass_HelloWorld_Penghitung.
   - Untuk filter topik, masukkan **`hello/world/counter/trigger`**.

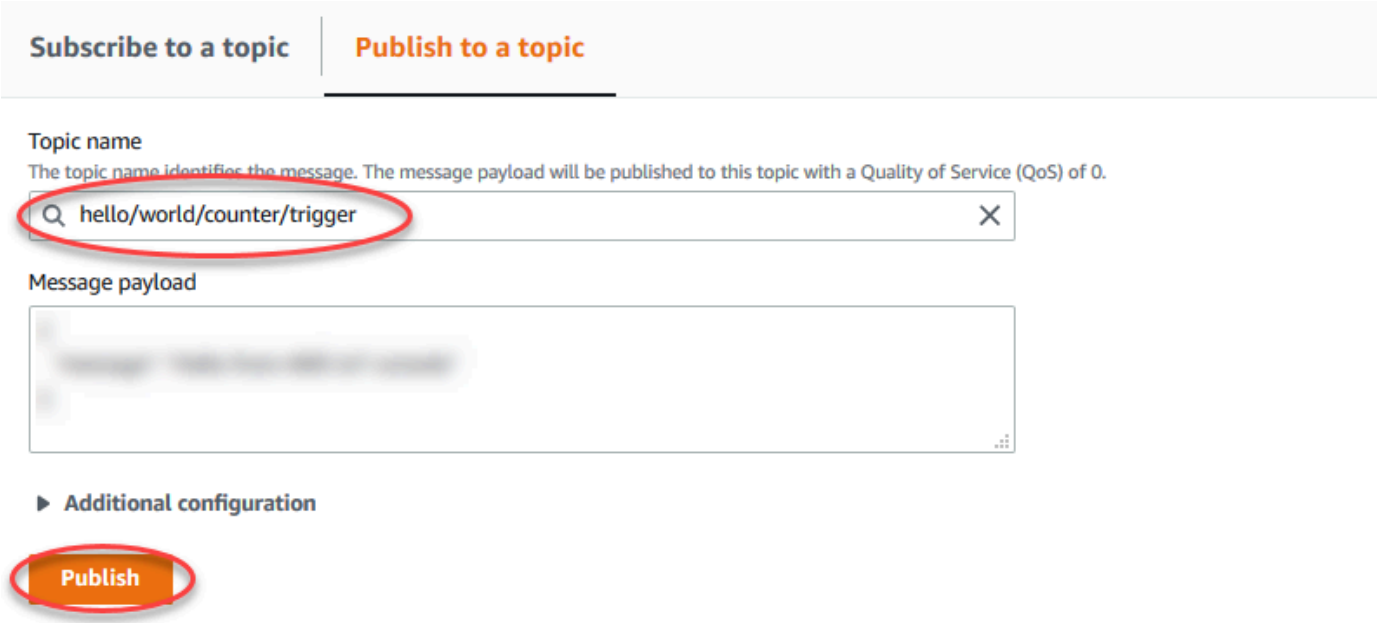
   Ekstensi `/trigger` digunakan dalam filter topik ini karena Anda membuat dua langganan dan tidak ingin mereka mengganggu satu sama lain.
7. Pastikan bahwa Greengrass daemon berjalan, seperti yang dijelaskan dalam Men-deploy konfigurasi cloud ke perangkat core.
8. Pada halaman konfigurasi grup, pilihDeploy.
9. Setelah deployment Anda selesai, kembali ke halaman beranda konsol AWS IoT dan pilih Uji.
10. Konfigurasi bidang berikut:

    - Untuk Topik langganan, masukkan **`hello/world/counter`**.
    - Untuk Kualitas Layanan, pilih 0.
    - Untuk Tampilan muatan MQTT, pilih Tampilkan muatan sebagai string.
11. Pilih Berlangganan.

    Tidak seperti Bagian 1 dari modul ini, Anda seharusnya tidak melihat pesan apa pun setelah Anda berlangganan `hello/world/counter`. Hal ini karena kode `greengrassHelloWorldCounter.py` yang menerbitkan ke topik `hello/world/counter` adalah di dalam fungsi handler, yang berjalan hanya ketika fungsi tersebut dipanggil.

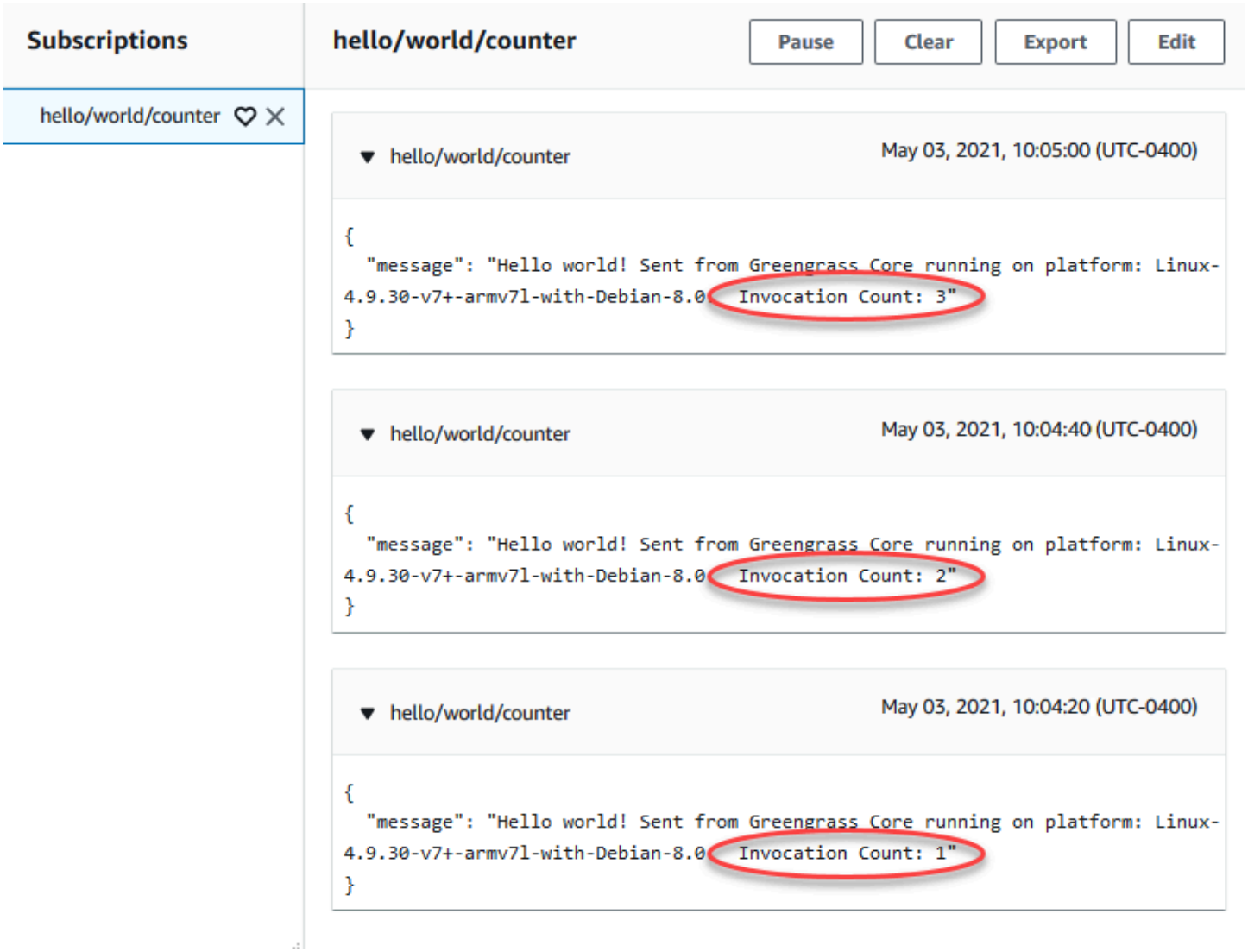
    Dalam modul ini, Anda mengonfigurasi `Greengrass_HelloWorld_Counter` fungsi Lambda akan dipanggil ketika menerima pesan MQTT pada `hello/world/counter/trigger` topik.

    ParameterGreengrass_HelloWorld_PenghitungkepadaIoT Cloudberlangganan memungkinkan fungsi untuk mengirim pesan keAWS IoTpada`hello/world/counter`Topik. ParameterIoT CloudkepadaGreengrass_HelloWorld_Penghitungberlangganan memungkinkanAWS IoTmengirim pesan ke fungsi`hello/world/counter/trigger`Topik.
12. Untuk menguji siklus hidup berumur panjang, memanggil fungsi Lambda dengan menerbitkan pesan ke `hello/world/counter/trigger` topik. Anda dapat menggunakan pesan default.



> **Note**
>
> Fungsi `Greengrass_HelloWorld_Counter` mengabaikan isi dari pesan yang diterima. Itu hanya menjalankan kode di `function_handler`, yang mengirim pesan ke `hello/world/counter` topik. Anda dapat meninjau kode ini dariAWS IoT GreengrassCore SDK for Pythondi atas GitHub.

Setiap kali pesan diterbitkan ke `hello/world/counter/trigger` topik, `my_counter` variabel bertambah. Jumlah doa ini ditampilkan dalam pesan yang dikirim dari fungsi Lambda. Karena fungsi handler termasuk siklus tidur 20 detik (`time.sleep(20)`), berulang kali memicu handler antrian tanggapan dari AWS IoT Greengrass Core.

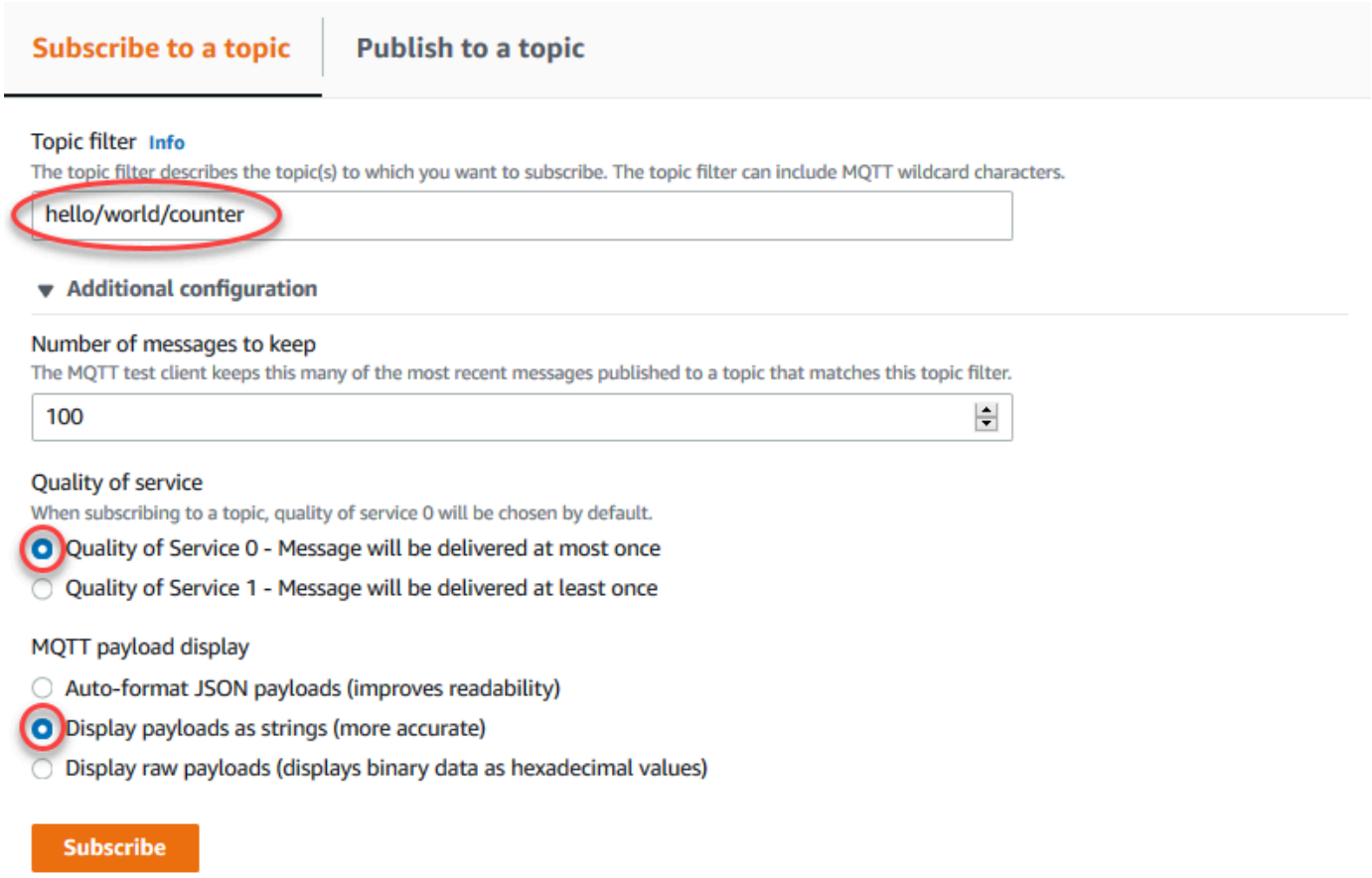


Uji fungsi Lambda sesuai permintaan

Sesuai permintaan fungsi Lambda mirip dalam fungsi untuk fungsi AWS Lambda berbasis cloud. Beberapa permohonan dari fungsi Lambda sesuai permintaan dapat berjalan secara paralel. Sebuah permohonan dari fungsi Lambda menciptakan kontainer terpisah untuk memproses permohonan atau menggunakan kembali kontainer yang ada, jika sumber daya mengizinkan. Setiap variabel atau preprocessing yang didefinisikan di luar fungsi handler tidak dipertahankan ketika kontainer dibuat.

1. Pada halaman konfigurasi grup, pilihFungsi Lambdatab.
2. Di bawahFungsi Lambda saya, pilihGreengrass_HelloWorld_CounterFungsi Lambda.
3. PadaGreengrass_HelloWorld_Counterhalaman detail, pilihedit.
4. UntukDipinned, pilihSalah, dan kemudian pilihSimpan.

5. Pada halaman konfigurasi grup, pilihDeploy.
6. Setelah deployment Anda selesai, kembali ke halaman beranda konsol AWS IoT dan pilih Uji.
7. Konfigurasi bidang berikut:
   - Untuk Topik langganan, masukkan **`hello/world/counter`**.
   - Untuk Kualitas Layanan, pilih 0.
   - Untuk Tampilan muatan MQTT, pilih Tampilkan muatan sebagai string.

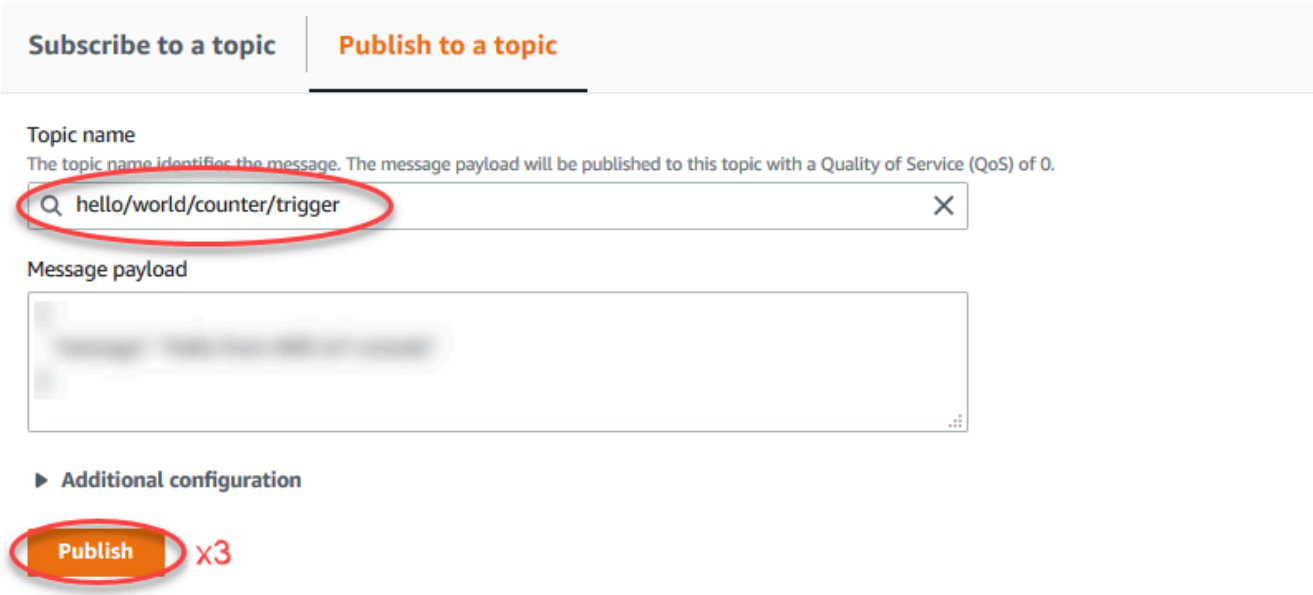


8. Pilih Berlangganan.

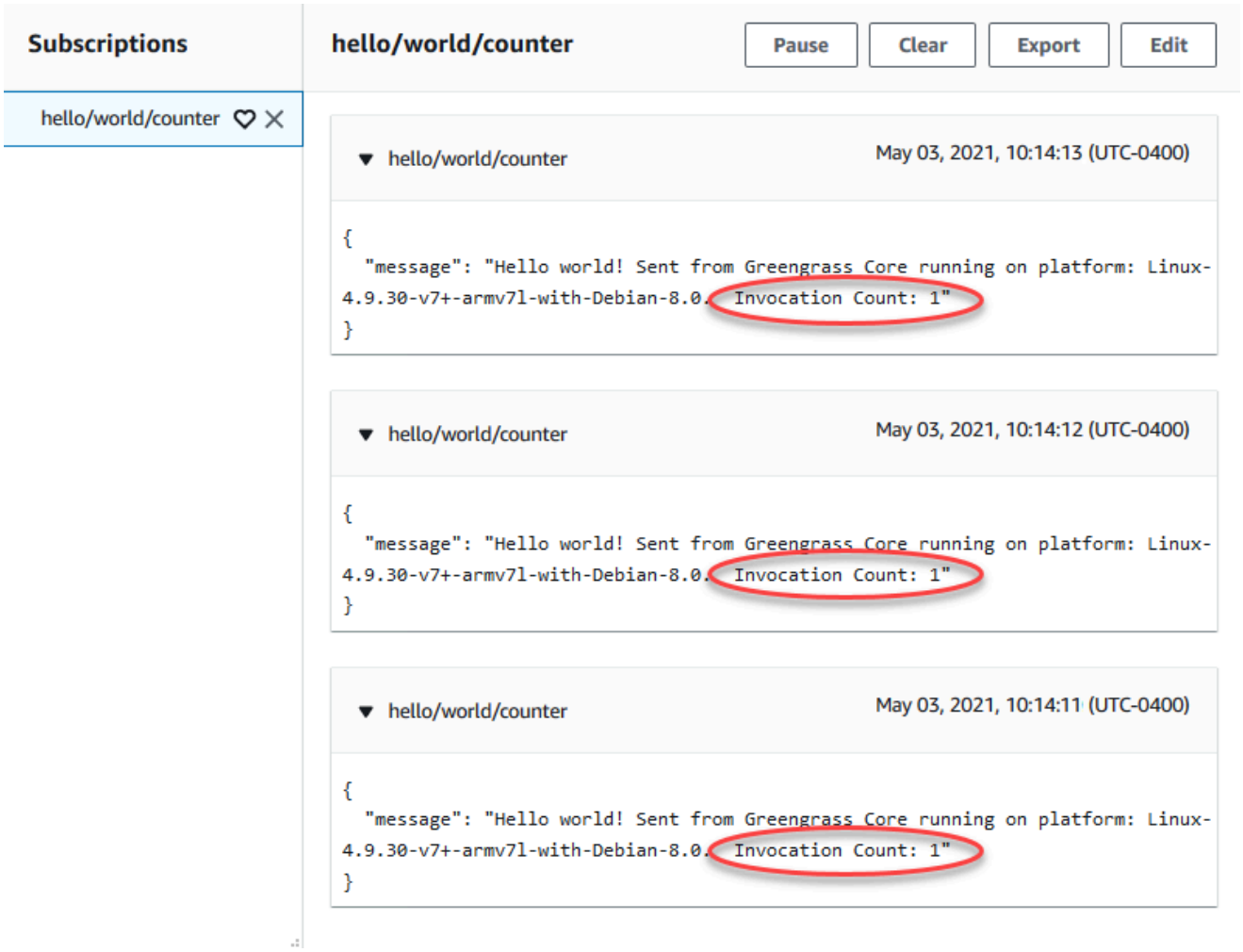
> **Note**
>
> Anda seharusnya tidak melihat pesan apa pun setelah Anda berlangganan.

9. Untuk menguji siklus hidup Sesuai Permintaan, meminta fungsi dengan menerbitkan pesan ke `hello/world/counter/trigger` topik. Anda dapat menggunakan pesan default.

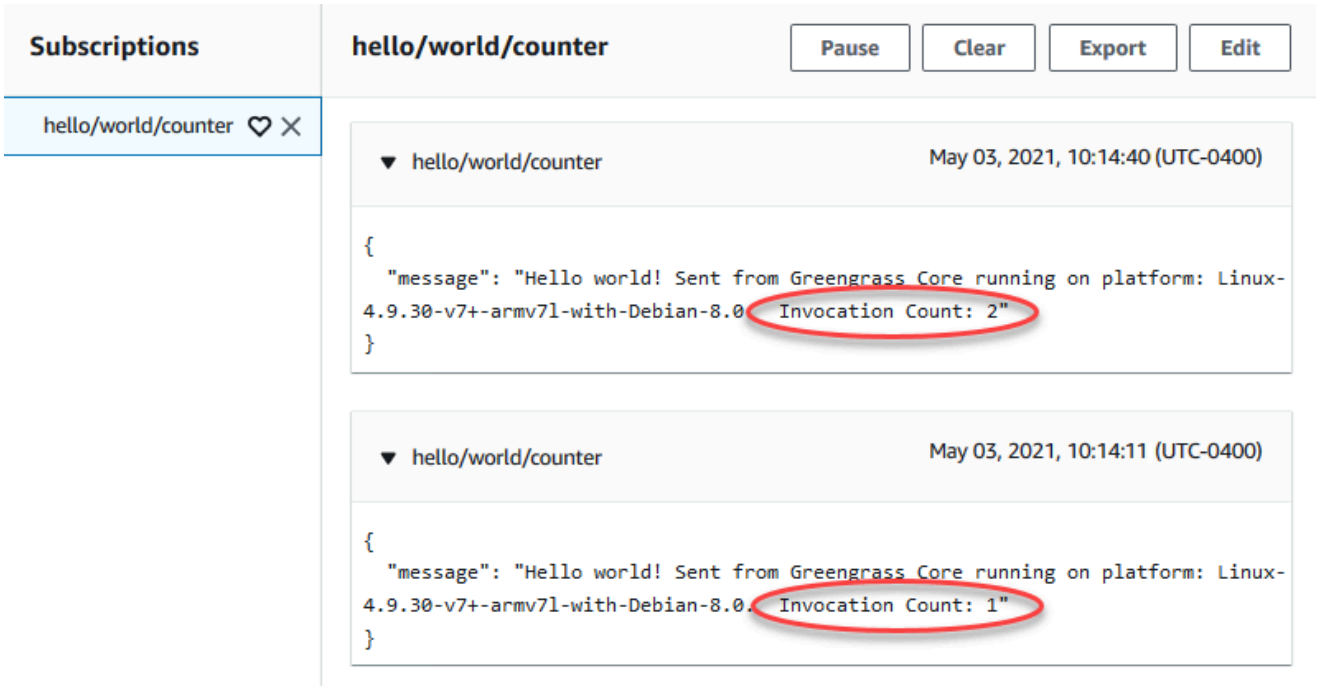
a. Pilih Terbitkan tiga kali dengan cepat, dalam waktu lima detik setiap tekan tombol.



Setiap terbitkan memanggil fungsi handler dan menciptakan sebuah kontainer untuk setiap permohonan. Jumlah permohonan tidak bertambah untuk tiga kali Anda memicu fungsi karena setiap fungsi Lambda on-demand memiliki kontainer/sandbox sendiri.



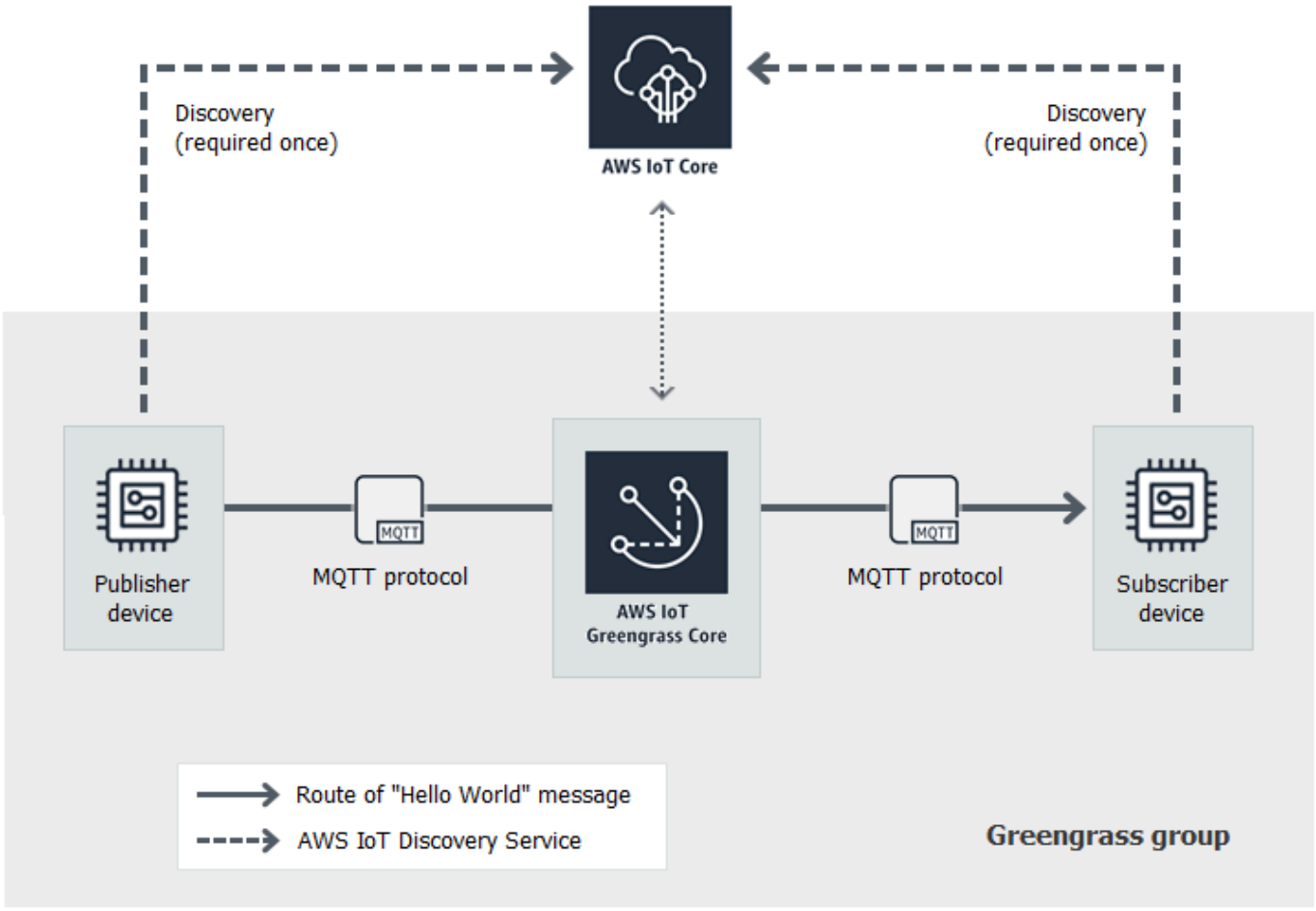
b. Setelah sekitar 30 detik, pilih Terbitkan ke topik. Jumlah permohonan harus bertambah menjadi 2. Hal ini menunjukkan bahwa kontainer dibuat dari permohonan sebelumnya sedang digunakan kembali, dan bahwa variabel preprocessing luar dari fungsi handler disimpan.



Anda sekarang harus memahami dua jenis fungsi Lambda yang dapat berjalan di AWS IoT Greengrass core. Modul berikutnya,Modul 4, menunjukkan kepada Anda bagaimana perangkat IoT lokal dapat berinteraksi dalamAWS IoT Greengrassgrup.

Modul 4: Berinteraksilah dengan perangkat klien dalamAWS IoT Greengrasskelompok

Modul ini menunjukkan kepada Anda bagaimana perangkat IoT lokal, dipanggilperangkat klienataupesawat, dapat terhubung ke dan berkomunikasi denganAWS IoT Greengrassperangkat inti. Perangkat klien yang terhubung keAWS IoT Greengrassinti adalah bagian dariAWS IoT Greengrasskelompok dan dapat berpartisipasi dalamAWS IoT Greengrassparadigma pemrograman. Dalam modul ini, satu perangkat klien mengirimkan pesan Hello World ke perangkat klien lain dalam grup Greengrass.



Sebelum Anda memulai, jalankan perintah pengaturan perangkat Greengrass skrip atau selesai Modul 1 dan Modul 2. Modul ini menciptakan dua perangkat klien simulasi. Anda tidak memerlukan komponen atau perangkat lain.

Modul ini akan memakan waktu kurang dari 30 menit untuk menyelesaikannya.

Topik

- Membuat perangkat klien dalamAWS IoT Greengrasskelompok
- Konfigurasi langganan
- Instal AWS IoT Device SDK untuk Python
- Uji komunikasi

Membuat perangkat klien dalamAWS IoT Greengrasskelompok

Pada langkah ini, Anda akan menambahkan dua perangkat klien ke grup Greengrass Anda. Proses ini termasuk mendaftarkan perangkat sebagaiAWS IoThal-hal dan mengonfigurasi sertifikat dan kunci untuk mengizinkan mereka untuk terhubung keAWS IoT Greengrass.

1. DiAWS IoTpanel navigasi konsol, di bawahKelola, PerluasPerangkat Greengrass, dan kemudian pilihGrup (V1).
2. Pilih grup target.
3. Pada halaman konfigurasi grup, pilihPerangkat klien, dan kemudian pilihAsosiasi.
4. DiMengaitkan perangkat klien dengan grup inimodal, pilihBuat baruAWS IoThal.

   ParameterMembuat hal-halhalaman terbuka di tab baru.
5. PadaMembuat hal-halhalaman, pilihMembuat hal tunggal, dan kemudian pilihSelanjutnya.
6. PadaTentukan properti halhalaman, daftarkan perangkat klien ini sebagai**`HelloWorld_Publisher`**, dan kemudian pilihSelanjutnya.
7. PadaKonfigurasi sertifikat perangkathalaman, pilihSelanjutnya.
8. PadaLampirkan kebijakan ke sertifikathalaman, lakukan salah satu hal berikut:
   - Pilih kebijakan yang ada yang memberikan izin yang diperlukan perangkat klien, lalu pilihMembuat hal.

     Modal terbuka di mana Anda dapat mengunduh sertifikat dan kunci yang digunakan perangkat untuk terhubung keAWS Clouddan intinya.
   - Buat dan lampirkan kebijakan baru yang memberikan izin perangkat klien. Lakukan hal berikut:

     a. Pilih Buat kebijakan.

        ParameterBuat kebijakanhalaman terbuka di tab baru.

     b. PadaBuat kebijakanhalaman, lakukan hal berikut:

        i. UntukNama kebijakan, memasukkan nama yang menjelaskan kebijakan, seperti**`GreengrassV1ClientDevicePolicy`**.

        ii. PadaKebijakan pernyataantab, di bawahDokumen kebijakan, pilihJSON.

        iii. Masukkan dokumen kebijakan berikut. Kebijakan ini memungkinkan perangkat klien untuk menemukan core Greengrass dan berkomunikasi pada semua

topik MQTT. Untuk informasi tentang cara membatasi akses kebijakan ini, lihatAutentikasi dan otorisasi perangkat untuk AWS IoT Greengrass.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Subscribe",
        "iot:Connect",
        "iot:Receive"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "greengrass:*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

iv. PilihBuatuntuk membuat kebijakan.

c. Kembali ke tab browser denganLampirkan kebijakan ke sertifikathalaman terbuka. Lakukan hal berikut:

i. DiKebijakandaftar, pilih kebijakan yang Anda buat, sepertiGreengrassV1ClientDevicePolicy.

Jika Anda tidak melihat kebijakan, pilih tombol refresh.

ii. PilihMembuat hal.

Modal terbuka di mana Anda dapat mengunduh sertifikat dan kunci yang digunakan perangkat untuk terhubung keAWS Clouddan intinya.

9. DiUnduh sertifikat dan kuncimodal, men-download sertifikat perangkat.

   > **Important**
   >
   > Sebelum Anda memilihSelesai, mengunduh sumber daya keamanan.

   Lakukan hal berikut:

   a. UntukSertifikat perangkat, pilihUnduhuntuk mengunduh sertifikat perangkat.

   b. UntukFile kunci publik, pilihUnduhuntuk mengunduh kunci publik untuk sertifikat.

   c. UntukFile kunci privat, pilihUnduhuntuk mengunduh file kunci privat untuk sertifikat.

   d. Review Autentikasi Server dalam AWS IoT Panduan Developer dan memilih root sertifikat CA yang sesuai. Kami merekomendasikan Anda menggunakan Amazon Trust Services (ATS) titik akhir dan ATS root sertifikat CA. Di bawahSertifikat CA, pilihUnduhuntuk sertifikat CA root.

   e. PilihSelesai.

   Membuat catatan ID sertifikat yang umum dalam nama file untuk sertifikat dan kunci perangkat. Anda membutuhkannya nanti.

10. Kembali ke tab browser denganMengaitkan perangkat klien dengan grup inimodal terbuka. Lakukan hal berikut:

    a. UntukAWS IoTNama Hal, pilihHelloWorld_Publisherhal yang Anda buat.

       Jika Anda tidak melihatnya, pilih tombol refresh.

    b. Pilih Kaitkan.

11. Ulangi langkah 3 - 10 untuk menambahkan perangkat klien kedua ke grup.

    Beri nama perangkat klien ini**`HelloWorld_Subscriber`**. Unduh sertifikat dan kunci untuk perangkat klien ini ke komputer Anda. Sekali lagi, buat catatan ID sertifikat yang umum dalam nama file untuk HelloWorldPerangkat _Subscriber.

    Sekarang Anda seharusnya memiliki dua perangkat klien di grup Greengrass Anda:

    - HelloWorld_Penerbit
    - HelloWorld_Subscriber

12. Buat folder di komputer Anda untuk kredensial keamanan perangkat klien ini. Salin sertifikat dan kunci ke dalam folder ini.

Konfigurasi langganan

Pada langkah ini, Anda mengaktifkan HelloWorld_Penerbit perangkat klien untuk mengirim pesan MQTT ke HelloWorld_Subscriber perangkat klien.

1. Pada halaman konfigurasi grup, pilihLangganantab, dan kemudian pilihTambahkan.
2. PadaBuat langgananhalaman, lakukan hal berikut untuk mengkonfigurasi langganan:

   a. UntukJenis Sumber, choosePerangkat klien, lalu pilihHelloWorld_Penerbit.

   b. Di bawahJenis target, choosePerangkat klien, lalu pilihHelloWorld_Pelanggan.

   c. Untuk Filter topik, masukkan **`hello/world/pubsub`**.

   > **Note**
   >
   > Anda dapat menghapus langganan dari modul sebelumnya. Pada kelompokLangganantab, pilih langganan untuk dihapus, lalu pilihHapus.

   d. Pilih Buat langganan.
3. Pastikan bahwa deteksi otomatis diaktifkan sehingga core Greengrass dapat menerbitkan daftar alamat IP-nya. Perangkat klien menggunakan informasi ini untuk menemukan core. Lakukan hal berikut:

   a. Pada halaman konfigurasi grup, pilihFungsi LambdaTab.

   b. Di bawahFungsi Lambda sistem, chooseDetektor IP, lalu pilihedit.

   c. DiMengedit pengaturan detektor IP, chooseSecara otomatis mendeteksi dan mengganti titik akhir broker MQTT, lalu pilihSimpan.
4. Pastikan bahwa Greengrass daemon berjalan, seperti yang dijelaskan dalam Men-deploy konfigurasi cloud ke perangkat core.
5. Pada halaman konfigurasi grup, pilihDeploy.

Status deployment ditampilkan di bawah nama grup pada header halaman. Untuk melihat detail deployment, pilihDeploymentTab.

Instal AWS IoT Device SDK untuk Python

Perangkat klien dapat menggunakanAWS IoT Device SDKuntuk Python untuk berkomunikasi denganAWS IoTdanAWS IoT Greengrassperangkat inti (menggunakan bahasa pemrograman Python). Untuk informasi lebih lanjut, termasuk persyaratan, lihatAWS IoT Device SDKuntuk PythonReadmedi atas GitHub.

Pada langkah ini, Anda menginstal SDK dan mendapatkan`basicDiscovery.py`fungsi sampel yang digunakan oleh perangkat klien simulasi pada komputer Anda.

1. Untuk menginstal SDK pada komputer anda, dengan semua komponen yang diperlukan, pilih sistem operasi Anda:

   Windows

   1. Buka prompt perintah yang ditinggikan dan jalankan perintah berikut:

      ```
      python --version
      ```

      Jika tidak ada informasi versi dikembalikan atau jika nomor versi kurang dari 2.7 untuk Python 2 atau kurang dari 3.3 untuk Python 3, ikuti petunjuk di Mengunduh Python untuk menginstal Python 2.7+ or Python 3.3+. Untuk informasi lebih lanjut, lihat Menggunakan Python pada Windows.
   2. Unduh AWS IoT Device SDK untuk Python sebagai `zip` file dan mengekstraksi ke lokasi yang sesuai pada komputer Anda.

      Buat catatan tentang path file ke `aws-iot-device-sdk-python-master` folder yang berisi `setup.py` file. Pada langkah selanjutnya, path file ini ditunjukkan oleh *`path-to-SDK-folder`*.
   3. Dari prompt perintah yang ditinggikan, jalankan yang berikut ini:

      ```
      cd path-to-SDK-folder
      python setup.py install
      ```

   macOS

   1. Buka jendela Terminal dan jalankan perintah berikut:

```
python --version
```

Jika tidak ada informasi versi dikembalikan atau jika nomor versi kurang dari 2.7 untuk Python 2 atau kurang dari 3.3 untuk Python 3, ikuti petunjuk di Mengunduh Python untuk menginstal Python 2.7+ or Python 3.3+. Untuk informasi lebih lanjut, lihat Menggunakan Python pada Macintosh.

2. Di jendela Terminal, jalankan perintah berikut untuk menentukan versi OpenSSL:

```
python
>>>import ssl
>>>print ssl.OPENSSL_VERSION
```

Buat catatan nilai versi OpenSSL.

> **Note**
>
> Jika Anda menjalankan Python 3, gunakan print(ssl.OPENSSL_VERSION).

Untuk menutup Python shell, jalankan perintah berikut:

```
>>>exit()
```

Jika versi OpenSSL adalah 1.0.1 atau yang lebih baru, lewati ke langkah c. Jika tidak, ikuti langkah-langkah berikut:

- Dari jendela Terminal, jalankan perintah berikut untuk menentukan apakah komputer menggunakan Simple Python Version Management:

  ```
  which pyenv
  ```

Jika path file dikembalikan, kemudian pilih tab Menggunakan **pyenv** ini. Jika tidak ada yang dikembalikan, pilih tab Tidak menggunakan **pyenv** ini.

Using pyenv

1. Lihat Phyton Releases for Mac OS X (atau serupa) untuk menentukan versi Python stabil terbaru. Pada contoh berikut, nilai ini ditunjukkan oleh *`versi terbaru Python`*.
2. Dari jendela Terminal, jalankan perintah berikut:

   ```
   pyenv install latest-Python-version
   pyenv global latest-Python-version
   ```

   Sebagai contoh, jika versi terbaru untuk Python 2 adalah 2.7.14, maka perintah ini adalah:

   ```
   pyenv install 2.7.14
   pyenv global 2.7.14
   ```

3. Tutup lalu buka kembali jendela Terminal lalu jalankan perintah berikut:

   ```
   python
   >>>import ssl
   >>>print ssl.OPENSSL_VERSION
   ```

   Versi OpenSSL harus setidaknya 1.0.1. Jika versi kurang dari 1.0.1, maka pembaruan gagal. Periksa nilai versi Python yang digunakan dalam pyenv install dan pyenv global perintahkan dan coba lagi.

4. Jalankan perintah berikut untuk keluar dari Python shell:

   ```
   exit()
   ```

Not using pyenv

1. Dari jendela Terminal, jalankan perintah berikut untuk menentukan apakah brew diinstal:

   ```
   which brew
   ```

   Jika path file tidak dikembalikan, instal `brew` sebagai berikut:

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/
Homebrew/install/master/install)"
```

> **Note**
>
> Ikuti petunjuk penginstalan. Unduh untuk alat baris perintah Xcode dapat memakan waktu lama.

2. Jalankan perintah berikut:

```
brew update
brew install openssl
brew install python@2
```

Untuk Python AWS IoT Device SDK membutuhkan OpenSSL versi 1.0.1 (atau yang lebih baru) dikompilasi dengan Python executable. Perintah brew install python menginstal sebuah `python2` executable yang memenuhi persyaratan ini. Executable `python2` diinstal di `/usr/local/bin` direktori, yang harus menjadi bagian dari `PATH` variabel lingkungan. Untuk mengonfirmasi, jalankan perintah berikut:

```
python2 --version
```

Jika `python2` informasi versi tersedia, lewati ke langkah selanjutnya. Jika tidak, secara permanen menambahkan `/usr/local/bin` jalur ke `PATH` dengan menambahkan baris berikut ke profil shell Anda:

```
export PATH="/usr/local/bin:$PATH"
```

Sebagai contoh, jika Anda menggunakan `.bash_profile` atau belum memiliki profil shell, jalankan perintah berikut dari jendela Terminal:

```
echo 'export PATH="/usr/local/bin:$PATH"' >> ~/.bash_profile
```

Selanjutnya, sumber profil shell Anda dan konfirmasikan bahwa `python2 --version` menyediakan informasi versi. Sebagai contoh, jika Anda menggunakan `.bash_profile`, jalankan perintah berikut:

```
source ~/.bash_profile
python2 --version
```

`python2` informasi versi harus dikembalikan.

3. Tambahkan baris berikut ke profil shell Anda:

```
alias python="python2"
```

Sebagai contoh, jika Anda menggunakan `.bash_profile` atau belum memiliki profil shell, jalankan perintah berikut:

```
echo 'alias python="python2"' >> ~/.bash_profile
```

4. Selanjutnya, sumber profil shell Anda. Sebagai contoh, jika Anda menggunakan `.bash_profile`, jalankan perintah berikut:

```
source ~/.bash_profile
```

Memanggil python menjalankan Python executable yang berisi versi OpenSSL yang dibutuhkan (`python2`) .

5. Jalankan perintah berikut:

```
python
 import ssl
 print ssl.OPENSSL_VERSION
```

Versi OpenSSL harus 1.0.1 atau yang lebih baru.

6. Untuk keluar dari Python shell, jalankan perintah berikut:

```
 exit()
```

3. Jalankan perintah berikut untuk menginstal AWS IoT Device SDK untuk Python:

```
cd ~
git clone https://github.com/aws/aws-iot-device-sdk-python.git
cd aws-iot-device-sdk-python
sudo python setup.py install
```

UNIX-like system

1. Dari jendela terminal, jalankan perintah berikut:

   ```
   python --version
   ```

   Jika tidak ada informasi versi dikembalikan atau jika nomor versi kurang dari 2.7 untuk Python 2 atau kurang dari 3.3 untuk Python 3, ikuti petunjuk di Mengunduh Python untuk menginstal Python 2.7+ or Python 3.3+. Untuk informasi lebih lanjut, lihat Menggunakan Python pada platform Unix.

2. Di terminal, jalankan perintah berikut untuk menentukan versi OpenSSL:

   ```
   python
   >>>import ssl
   >>>print ssl.OPENSSL_VERSION
   ```

   Buat catatan nilai versi OpenSSL.

   > **Note**
   >
   > Jika Anda menjalankan Python 3, gunakan print(ssl.OPENSSL_VERSION).

   Untuk menutup Python shell, jalankan perintah berikut:

   ```
   exit()
   ```

   Jika versi OpenSSL adalah 1.0.1 atau yang lebih baru, lewati ke langkah selanjutnya. Jika tidak, jalankan perintah untuk memperbarui OpenSSL untuk distribusi Anda (sebagai contoh, `sudo yum update openssl`, `sudo apt-get update`, dan sebagainya).

Konfirmasikan bahwa versi OpenSSL adalah 1.0.1 atau yang lebih baru dengan menjalankan perintah berikut:

```
python
>>>import ssl
>>>print ssl.OPENSSL_VERSION
>>>exit()
```

3. Jalankan perintah berikut untuk menginstal AWS IoT Device SDK untuk Python:

```
cd ~
git clone https://github.com/aws/aws-iot-device-sdk-python.git
cd aws-iot-device-sdk-python
sudo python setup.py install
```

2. Setelah AWS IoT Device SDK untuk Python yang diinstal, arahkan ke `samples` folder dan buka `greengrass` folder.

   Untuk tutorial ini, Anda menyalin `basicDiscovery.py` fungsi sampel yang menggunakan sertifikat dan kunci yang diunduh di the section called “Membuat perangkat klien dalamAWS IoT Greengrasskelompok”.

3. Salin`basicDiscovery.py`ke folder yang berisi HelloWorld_Publisher dan HelloWorld_Subscriber sertifikat dan kunci perangkat.

Uji komunikasi

1. Pastikan bahwa komputer Anda dan AWS IoT Greengrass perangkat core terhubung ke internet menggunakan jaringan yang sama.

   a. Pada AWS IoT Greengrass perangkat core, jalankan perintah berikut untuk menemukan alamat IP-nya.

   ```
   hostname -I
   ```

   b. Dalam komputer Anda, jalankan perintah berikut menggunakan alamat IP core. Anda dapat menggunakan Ctrl + C untuk menghentikan perintah ping ini.

   ```
   ping IP-address
   ```

Output yang mirip dengan berikut ini menunjukkan sukses komunikasi antara komputer dan AWS IoT Greengrass perangkat core (0% packet loss):

```
$ping 176.32.103.205
PING 176.32.103.205 (176.32.103.205) 56(84) bytes of data.
64 bytes from 176.32.103.205: icmp_seq=1 ttl=230 time=77.2 ms
64 bytes from 176.32.103.205: icmp_seq=2 ttl=230 time=77.1 ms
64 bytes from 176.32.103.205: icmp_seq=3 ttl=230 time=77.1 ms
64 bytes from 176.32.103.205: icmp_seq=4 ttl=230 time=77.1 ms
64 bytes from 176.32.103.205: icmp_seq=5 ttl=230 time=77.1 ms
64 bytes from 176.32.103.205: icmp_seq=6 ttl=230 time=77.1 ms
^C
--- 176.32.103.205 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5549ms
rtt min/avg/max/mdev = 77.107/77.172/77.256/0.361 ms
```

> **Note**
>
> Jika Anda tidak dapat melakukan ping ke instans EC2 yang sedang menjalankan AWS IoT Greengrass, pastikan bahwa aturan inbound grup keamanan pada instans mengizinkan lalu lintas ICMP untuk pesan permintaan Echo ini. Untuk informasi lebih lanjut, lihat Penambahan aturan ke sebuah grup keamanan dalam Panduan Pengguna Amazon EC2 untuk Instans Linux.
>
> Pada komputer host Windows, di Windows Firewall dengan aplikasi Keamanan Lanjutan, Anda mungkin juga perlu mengaktifkan aturan masuk yang mengizinkan permintaan inbound echo (sebagai contoh, Berbagi File dan Printer (Permintaan Echo - ICMPv4-In)), atau buat satu.

2. Dapatkan titik akhir AWS IoT Anda.

   a. DariAWS IoTkonsolpanel navigasi, pilihPengaturan.

   b. Di bawahEndpoint data perangkat, buat catatan dari nilaiTitik akhir. Anda menggunakan nilai ini untuk mengganti *AWS_IOT_ENDPOINT* placeholder di perintah dalam langkah-langkah berikut.

   > **Note**
   >
   > Pastikan bahwa titik akhir Anda sesuai dengan jenis sertifikat Anda.

3. Pada komputer Anda (bukan AWS IoT Greengrass perangkat core), buka dua jendela baris perintah (terminal atau prompt perintah). Satu jendela mewakili HelloWorldPerangkat klien _Publisher dan yang lainnya mewakili HelloWorld_Subscriber perangkat klien.

   Setelah eksekusi, `basicDiscovery.py` mencoba untuk mengumpulkan informasi tentang lokasi AWS IoT Greengrass core pada titik akhirnya. Informasi ini disimpan setelah perangkat klien ditemukan dan berhasil terhubung ke core. Ini mengizinkan pengiriman pesan dan operasi di masa mendatang dijalankan secara lokal (tanpa memerlukan koneksi internet).

   > **Note**
   >
   > ID klien yang digunakan untuk koneksi MQTT harus sesuai dengan nama hal dari perangkat klien. Skrip `basicDiscovery.py` mengatur ID klien untuk koneksi MQTT ke nama hal yang Anda tentukan ketika Anda menjalankan skrip.
   > Jalankan perintah berikut dari folder yang berisi `basicDiscovery.py` file untuk informasi penggunaan skrip terperinci:
   >
   > ```
   > python basicDiscovery.py --help
   > ```

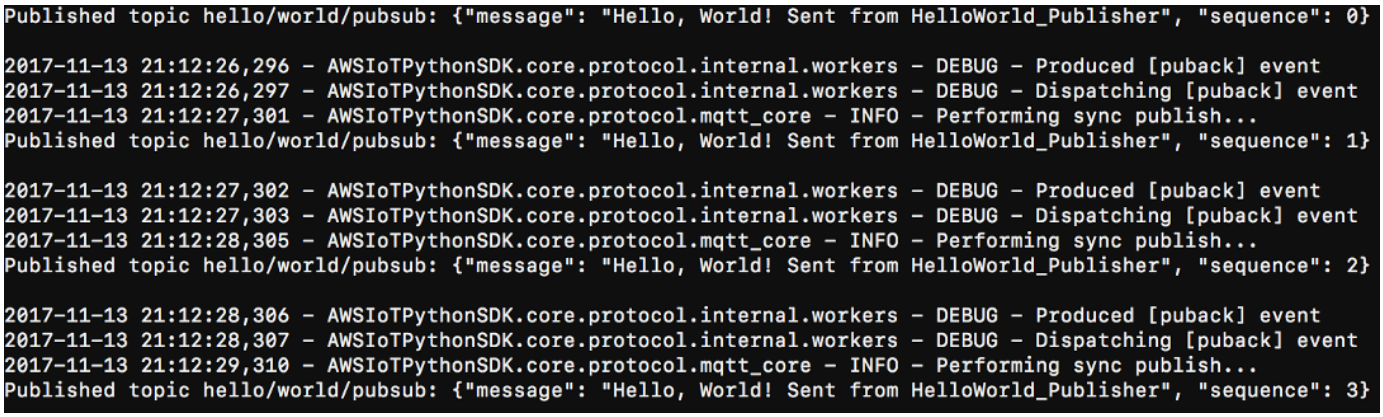
4. Dari HelloWorldJendela perangkat klien _Publisher, jalankan perintah berikut.

   - Ganti*path-to-certs-folder*dengan jalur ke folder yang berisi sertifikat, kunci, dan`basicDiscovery.py`.
   - Ganti *AWS_IOT_ENDPOINT* dengan titik akhir Anda.
   - Ganti keduanya*publisherCertId*instance dengan ID sertifikat dalam nama file untuk Anda HelloWorld_Penerbit perangkat klien.

   ```
   cd path-to-certs-folder
   python basicDiscovery.py --endpoint AWS_IOT_ENDPOINT --rootCA AmazonRootCA1.pem
    --cert publisherCertId-certificate.pem.crt --key publisherCertId-private.pem.key
    --thingName HelloWorld_Publisher --topic 'hello/world/pubsub' --mode publish --
   message 'Hello, World! Sent from HelloWorld_Publisher'
   ```

   Anda akan melihat output yang serupa dengan yang berikut, yang mencakup entri seperti `Published topic 'hello/world/pubsub': {"message": "Hello, World! Sent from HelloWorld_Publisher", "sequence": 1}`.

> **Note**
>
> Jika skrip mengembalikan `error: unrecognized arguments` pesan, mengubah tanda kutip tunggal untuk ganda tanda kutip untuk `--topic` dan `--message` parameter dan jalankan perintah lagi.
> Untuk memecahkan masalah koneksi, Anda dapat mencoba menggunakan Deteksi IP manual.

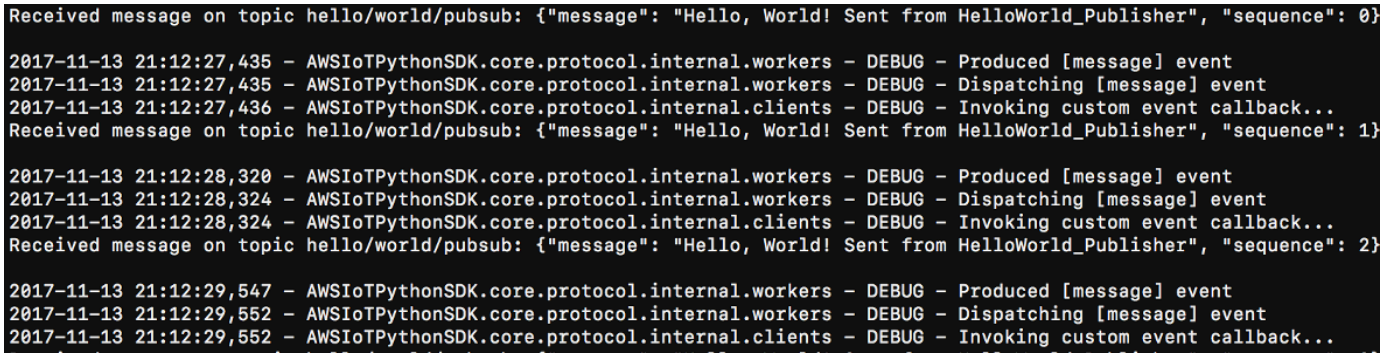


5. Dari HelloWorldJendela perangkat klien _Subscriber, jalankan perintah berikut.

   - Ganti*path-to-certs-folder*dengan jalur ke folder yang berisi sertifikat, kunci, dan`basicDiscovery.py`.
   - Ganti *AWS_IOT_ENDPOINT* dengan titik akhir Anda.
   - Ganti keduanya*subscriberCertId*instance dengan ID sertifikat dalam nama file untuk Anda HelloWorld_Subscriber perangkat klien.

```
cd path-to-certs-folder
python basicDiscovery.py --endpoint AWS_IOT_ENDPOINT --rootCA AmazonRootCA1.pem --
cert subscriberCertId-certificate.pem.crt --key subscriberCertId-private.pem.key --
thingName HelloWorld_Subscriber --topic 'hello/world/pubsub' --mode subscribe
```

   Anda harus melihat output berikut, yang mencakup entri seperti `Received message on topic hello/world/pubsub: {"message": "Hello, World! Sent from HelloWorld_Publisher", "sequence": 1}`.



TutupHelloWorld_Publisherjendela untuk menghentikan pesan dari diperoleh diHelloWorld_Subscriberjendela.

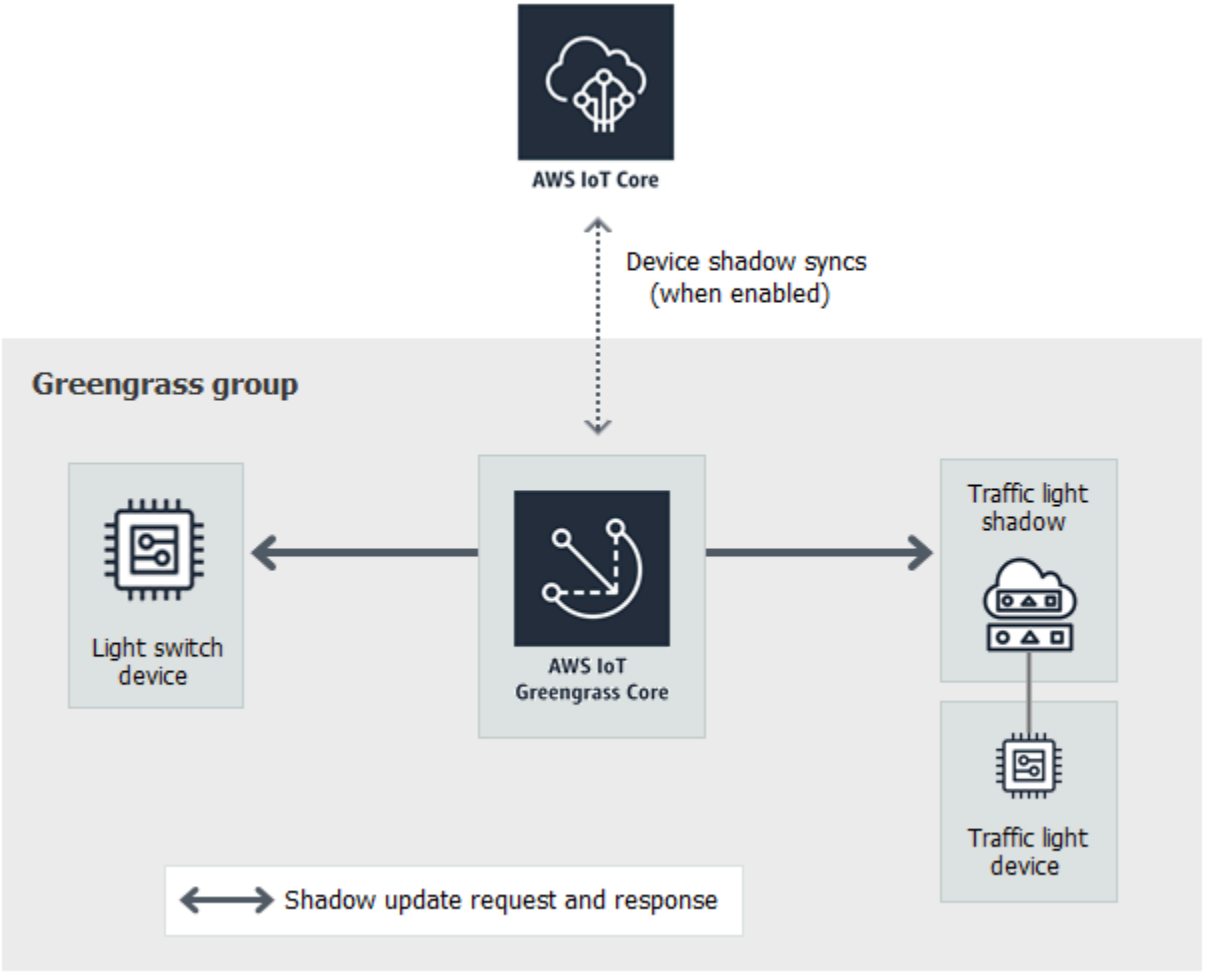
Pengujian pada jaringan perusahaan mungkin mengganggu koneksi ke core. Sebagai solusi, Anda dapat secara manual memasukkan titik akhir. Hal ini memastikan bahwa `basicDiscovery.py` skrip terhubung ke alamat IP yang benar dari AWS IoT Greengrass perangkat core.

Untuk memasukkan titik akhir secara manual

1. DiAWS IoTpanel navigasi konsol, di bawahKelola, perluasPerangkat Greengrass, lalu pilihGrup (V1).
2. Di bawah grup Greengrass, pilih grup Anda.
3. Konfigurasikan inti untuk secara otomatis mengelola titik akhir broker MQTT. Lakukan hal berikut:
   a. Pada halaman konfigurasi grup, pilihFungsi Lambdatab.
   b. Di bawahFungsi Lambda sistemchooseDetektor IP, lalu pilihEdit.
   c. DiEdit pengaturan detektor IPchooseKelola endpoint broker MQTT secara manual, lalu pilihSimpan.
4. Masukkan endpoint broker MQTT untuk inti. Lakukan hal berikut:
   a. Di bawahIkhtisar, chooseCore Greengrass.
   b. Di bawahEndpoint broker MQTTchooseMengelola titik akhir.
   c. MemiilihTambahkan titik akhirdan pastikan bahwa Anda hanya memiliki satu nilai titik akhir. Nilai ini harus menjadi titik akhir alamat IP untuk port 8883 dari AWS IoT Greengrass perangkat core (sebagai contoh, `192.168.1.4`).
   d. Pilih Pembaruan.

Modul 5: Berinteraksi dengan bayangan perangkat

Modul lanjutan ini menunjukkan cara Anda dapat berinteraksi dengan AndaAWS IoTbayangan perangkatdalam sebuahAWS IoT Greengrassgrup. Bayangan adalah dokumen JSON yang digunakan untuk menyimpan informasi keadaan saat ini atau yang diinginkan untuk suatu hal. Dalam modul ini, Anda menemukan bagaimana satu perangkat klien (`GG_Switch`) dapat memodifikasi keadaan perangkat klien lain (`GG_TrafficLight`) dan bagaimana negara-negara ini dapat disinkronkan keAWS IoT Greengrasscloud:



Sebelum Anda memulai, jalankan skrip Pengaturan perangkat Greengrass ini, atau pastikan bahwa Anda telah menyelesaikan Modul 1 dan Modul 2. Anda juga harus memahami cara menghubungkan perangkat klien keAWS IoT Greengrassinti (Modul 4). Anda tidak memerlukan komponen atau perangkat lain.

Modul ini akan memakan waktu sekitar 30 menit untuk menyelesaikannya.

Topik

- Konfigurasikan perangkat dan langganan
- Unduh file yang diperlukan
- Uji komunikasi (sinkronisasi perangkat dinonaktifkan)
- Uji komunikasi (sinkronisasi perangkat diaktifkan)

Konfigurasikan perangkat dan langganan

Bayangan dapat disinkronkan ke AWS IoT ketika AWS IoT Greengrass core terhubung ke internet. Dalam modul ini, Anda pertama kali menggunakan bayangan lokal tanpa menyinkronkan ke cloud. Kemudian, Anda mengaktifkan sinkronisasi cloud.

Setiap perangkat klien memiliki bayangannya sendiri. Untuk informasi lebih lanjut, lihat Layanan bayangan perangkat untuk AWS IoT di AWS IoT Panduan Developer.

1. Pada halaman konfigurasi grup, pilihPerangkat klienTab.
2. DariPerangkat klientab, tambahkan dua perangkat klien baru diAWS IoT Greengrassgrup. Untuk langkah-langkah rinci tentang proses ini, lihat the section called “Membuat perangkat klien dalamAWS IoT Greengrasskelompok”.
   - Beri nama perangkat klien**`GG_Switch`**dan**`GG_TrafficLight`**.
   - Hasilkan dan unduh sumber daya keamanan untuk kedua perangkat klien.
   - Buat catatan ID sertifikat dalam nama file sumber daya keamanan untuk perangkat klien. Anda menggunakan nilai-nilai ini kemudian.
3. Buat folder di komputer Anda untuk kredensyal keamanan perangkat klien. Salin sertifikat dan kunci ke dalam folder ini.
4. Pastikan bahwa perangkat klien diatur untuk menggunakan bayangan lokal dan tidak disinkronkan denganAWS Cloud. Jika tidak, pilih perangkat klien, pilihSinkronisasi bayangan, dan kemudian pilihNonaktifkan sinkronisasi bayangan dengan cloud.
5. Tambahkan langganan dalam tabel berikut ke grup Anda. Sebagai contoh, untuk membuat langganan pertama:
   a. Pada halaman konfigurasi grup, pilihLangganantab, dan kemudian pilihTambahkan.
   b. UntukJenis Sumber, pilihPerangkat klien, dan kemudian pilihG_Switch.
   c. UntukJenis target, pilihLayanan, dan kemudian pilihLayanan Bayangan Lokal.

d. Untuk Filter topik, masukkan **`$aws/things/GG_TrafficLight/shadow/update`**

e. Pilih Buat langganan.

Topik harus dimasukkan persis seperti yang ditunjukkan pada tabel. Meskipun mengizinkan untuk menggunakan wildcard untuk mengonsolidasikan beberapa langganan, kami tidak menyarankan praktik ini. Untuk informasi lebih lanjut, lihat topik Bayangan MQTT di AWS IoT Panduan Developer.

Sumber	Target	Topik	Catatan
G_Switch	Layanan Bayangan Lokal	$aws/things/GGGGGGG_TrafficLight/bayangan/update	GG_Switch mengirimkan permintaan pembaruan untuk memperbarui topik.
Layanan Bayangan Lokal	G_Switch	$aws/things/GGGGGGG_TrafficLight/shadow/update/accepted	GG_Switch butuh mengetahui apakah permintaan pembaruan diterima.
Layanan Bayangan Lokal	G_Switch	$aws/things/GGGGGGG_TrafficLight/shadow/update/rejected	GG_Switch perlu mengetahui apakah permintaan pembaruan ditolak.
GG_TrafficLight	Layanan Bayangan Lokal	$aws/things/GGGGGGG_TrafficLight/bayangan/update	GG_TrafficLight mengirimkan pembaruan keadaannya untuk pembaruan topik.

Sumber	Target	Topik	Catatan
Layanan Bayangan Lokal	GG_TrafficLight	$aws/things/GGGGGGG_TrafficLight/shadow/update/delta	Layanan bayangan lokal mengirimkan pembaruan yang diterima untuk GGGGGGGGGGGTrafficLight melalui topik delta.
Layanan Bayangan Lokal	GG_TrafficLight	$aws/things/GGGGGGG_TrafficLight/shadow/update/accepted	GG_TrafficLight perlu mengetahui apakah pembaruan keadaannya diterima.
Layanan Bayangan Lokal	GG_TrafficLight	$aws/things/GGGGGGG_TrafficLight/shadow/update/rejected	GG_TrafficLight perlu mengetahui apakah pembaruan keadaannya ditolak.

Langganan baru ditampilkan diLanggananTab.

> **Note**
>
> Untuk informasi tentang $ karakter, lihat topik yang Dipesan.

6. Pastikan bahwa deteksi otomatis diaktifkan sehingga core Greengrass dapat menerbitkan daftar alamat IP-nya. Perangkat klien menggunakan informasi ini untuk menemukan core. Lakukan hal berikut:

   a. Pada halaman konfigurasi grup, pilihFungsi LambdaTab.

   b. Di bawahFungsi Lambda sistem, pilihDetektor IP, dan kemudian pilihedit.

   c. DiPengaturan detektor IP, pilihSecara otomatis mendeteksi dan mengganti titik akhir broker MQTT, dan kemudian pilihSimpan.

7. Pastikan bahwa Greengrass daemon berjalan, seperti yang dijelaskan dalam Men-deploy konfigurasi cloud ke perangkat core.

8. Pada halaman konfigurasi grup, pilihDeploy.

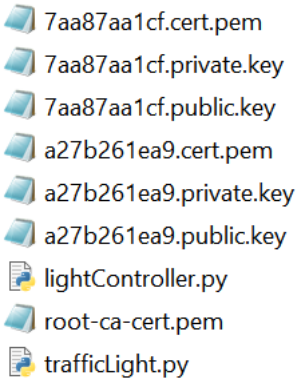
Unduh file yang diperlukan

1. Jika Anda belum melakukannya, instal AWS IoT Device SDK untuk Phyton. Untuk petunjuk, lihat langkah 1 dalam the section called “Instal AWS IoT Device SDK untuk Python”.

   SDK ini digunakan oleh perangkat klien untuk berkomunikasi denganAWS IoTdan denganAWS IoT Greengrassperangkat core.

2. Dari TrafficLightfolder contoh pada GitHub, unduh`lightController.py`dan`trafficLight.py`file ke komputer Anda. Menyempannya dalam folder yang berisi GGG_TrafficLight sertifikat dan kunci perangkat klien.

   Parameter`lightController.py`script sesuai dengan perangkat klien GG_Switch, dan`trafficLight.py`script sesuai dengan GG_TrafficLight perangkat klien.

   > **Note**
   >
   > Contoh file Python disimpan dalam AWS IoT Greengrass Core SDK for Python repositori untuk kenyamanan, tetapi mereka tidak menggunakan AWS IoT Greengrass Core SDK.

Uji komunikasi (sinkronisasi perangkat dinonaktifkan)

1. Pastikan bahwa komputer Anda dan AWS IoT Greengrass perangkat core terhubung ke internet menggunakan jaringan yang sama.

a. Pada AWS IoT Greengrass perangkat core, jalankan perintah berikut untuk menemukan alamat IP-nya.

```
hostname -I
```

b. Dalam komputer Anda, jalankan perintah berikut menggunakan alamat IP core. Anda dapat menggunakan Ctrl + C untuk menghentikan perintah ping ini.

```
ping IP-address
```

Output yang mirip dengan berikut ini menunjukkan sukses komunikasi antara komputer dan AWS IoT Greengrass perangkat core (0% packet loss):

```
$ping 176.32.103.205
PING 176.32.103.205 (176.32.103.205) 56(84) bytes of data.
64 bytes from 176.32.103.205: icmp_seq=1 ttl=230 time=77.2 ms
64 bytes from 176.32.103.205: icmp_seq=2 ttl=230 time=77.1 ms
64 bytes from 176.32.103.205: icmp_seq=3 ttl=230 time=77.1 ms
64 bytes from 176.32.103.205: icmp_seq=4 ttl=230 time=77.1 ms
64 bytes from 176.32.103.205: icmp_seq=5 ttl=230 time=77.1 ms
64 bytes from 176.32.103.205: icmp_seq=6 ttl=230 time=77.1 ms
^C
--- 176.32.103.205 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5549ms
rtt min/avg/max/mdev = 77.107/77.172/77.256/0.361 ms
```

> **Note**
>
> Jika Anda tidak dapat melakukan ping ke instans EC2 yang sedang menjalankan AWS IoT Greengrass, pastikan bahwa aturan inbound grup keamanan pada instans mengizinkan lalu lintas ICMP untuk pesan permintaan Echo ini. Untuk informasi lebih lanjut, lihat Penambahan aturan ke sebuah grup keamanan dalam Panduan Pengguna Amazon EC2 untuk Instans Linux.
>
> Pada komputer host Windows, di Windows Firewall dengan aplikasi Keamanan Lanjutan, Anda mungkin juga perlu mengaktifkan aturan masuk yang mengizinkan permintaan inbound echo (sebagai contoh, Berbagi File dan Printer (Permintaan Echo - ICMPv4-In)), atau buat satu.

2. Dapatkan titik akhir AWS IoT Anda.

a. DariAWS IoTkonsolpanel navigasi, pilihPengaturan.

b. Di bawahTitik akhir data perangkat perangkat, buat catatan dari nilaiTitik akhir. Anda menggunakan nilai ini untuk mengganti *AWS_IOT_ENDPOINT* placeholder di perintah dalam langkah-langkah berikut.

> **Note**
>
> Pastikan bahwa titik akhir Anda sesuai dengan jenis sertifikat Anda.

3. Pada komputer Anda (bukan AWS IoT Greengrass perangkat core), buka dua jendela baris perintah (terminal atau prompt perintah). Satu jendela mewakili perangkat klien GG_Switch GG_Switch dan yang lainnya mewakili GGG_SwitchTrafficLight perangkat klien.

   a. Dari jendela perangkat klien GGG_Switch, jalankan perintah berikut.

   - Ganti*path-to-certs-folder*dengan jalur ke folder yang berisi sertifikat, kunci, dan file Python.
   - Ganti *AWS_IOT_ENDPOINT* dengan titik akhir Anda.
   - Ganti keduanya*switchCertId*instans dengan ID sertifikat dalam nama file untuk perangkat klien GG_Switch Anda.

```
cd path-to-certs-folder
python lightController.py --endpoint AWS_IOT_ENDPOINT --rootCA
 AmazonRootCA1.pem --cert switchCertId-certificate.pem.crt --key switchCertId-
private.pem.key --thingName GG_TrafficLight --clientId GG_Switch
```

   b. Dari GGGGGG_TrafficLight jendela perangkat klien, jalankan perintah berikut.

   - Ganti*path-to-certs-folder*dengan jalur ke folder yang berisi sertifikat, kunci, dan file Python.
   - Ganti *AWS_IOT_ENDPOINT* dengan titik akhir Anda.
   - Ganti keduanya*lightCertId*contoh dengan ID sertifikat dalam nama file untuk GG_ AndaTrafficLight perangkat klien.

```
cd path-to-certs-folder
```

```
python trafficLight.py --endpoint AWS_IOT_ENDPOINT --rootCA AmazonRootCA1.pem
 --cert lightCertId-certificate.pem.crt --key lightCertId-private.pem.key --
thingName GG_TrafficLight --clientId GG_TrafficLight
```

Setiap 20 detik, switch memperbarui keadaan bayangan untuk G, Y, dan R, dan light menampilkan keadaan baru, seperti yang ditunjukkan selanjutnya.

Output GG_Switch:

```
{"state":{"desired":{"property":"R"}}}
2018-12-20 12:23:01,446 - AWSIoTPythonSDK.core.protocol.mqtt_core - INFO - Performing sync publish...
~~~~~~~~~~~~Shadow Update Accepted~~~~~~~~~~~~~~~
Update request with token: 3b22e27c-930d-4c6a-8562-9f86088249f4 accepted!
property: R
~~~~~~~~~~~~~~~~~~~~~~~~~~
```

GG_TrafficLight Output:

```
++++++++ Received Shadow Delta ++++++++++
{u'state': {u'property': u'R'}, u'metadata': {u'property': {u'timestamp': 1545337381}}, u'version': 33, u'clientToken':
u'3b22e27c-930d-4c6a-8562-9f86088249f4'}
property: R
version: 33
++++++++++++++++++++++


Light changed to: R
{"state":{"reported":{"property":"R"}}}
2018-12-20 12:23:01,539 - AWSIoTPythonSDK.core.protocol.mqtt_core - INFO - Performing sync publish...
~~~~~~~~~~~~ Shadow Update Accepted ~~~~~~~~~~~~~~~
Update request with token: f552109f-c1c2-4ae6-a841-8443506eefcb accepted!
property: R
~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Saat dijalankan untuk pertama kalinya, setiap skrip perangkat klien menjalankanAWS IoT Greengrasslayanan penemuan untuk terhubung keAWS IoT Greengrassinti (melalui internet). Setelah perangkat klien ditemukan dan berhasil tersambung ke perangkat klienAWS IoT Greengrasscore, operasi future dapat dijalankan secara lokal.

> **Note**
>
> Skrip `lightController.py` dan `trafficLight.py` menyimpan informasi koneksi di `groupCA` folder, yang dibuat dalam folder yang sama sebagai skrip. Jika Anda menerima eror koneksi, pastikan bahwa alamat IP di file `ggc-host` cocok dengan titik akhir alamat IP untuk core Anda.

4. DiAWS IoTkonsol, pilihAWS IoT Greengrasskelompok, pilihPerangkat klientab, dan kemudian pilihGG_TrafficLightuntuk membuka perangkat klienAWS IoThalaman detail hal.

5. PilihBayangan Perangkattab. Setelah GG_Switch mengubah negara, seharusnya tidak ada pembaruan untuk bayangan ini. Itu karena GG_TrafficLight diatur keNonaktifkan sinkronisasi bayangan dengan cloud.
6. tekan tekanCtrl+Cdi GG_Switch (`lightController.py`) jendela perangkat klien. Anda harus melihat bahwa GG_TrafficLight (`trafficLight.py`) jendela berhenti menerima pesan perubahan negara.

   Jaga jendela ini terbuka sehingga Anda dapat menjalankan perintah di bagian selanjutnya.

Uji komunikasi (sinkronisasi perangkat diaktifkan)

Untuk uji ini, Anda mengonfigurasi GGGGG_TrafficLight bayangan perangkat untuk disinkronkanAWS IoT. Anda menjalankan perintah yang sama seperti pada pengujian sebelumnya, tetapi kali ini status bayangan di cloud diperbarui ketika GG_Switch mengirimkan permintaan pembaruan.

1. DiAWS IoTkonsol, pilihAWS IoT Greengrasskelompok, dan kemudian memilihPerangkat klienTab.
2. Pilih GGGGG_TrafficLight perangkat, pilihSinkronisasi bayangan, dan kemudian pilihAktifkan sinkronisasi bayangan dengan cloud.

   Anda harus menerima notifikasi bahwa keadaan sinkronisasi bayangan perangkat telah diperbarui.
3. Pada halaman konfigurasi grup, pilihDeploy.
4. Dalam dua jendela baris perintah, jalankan perintah dari uji sebelumnya untukG_SwitchdanGG_TrafficLightperangkat klien.
5. Sekarang, periksa keadaan bayangan di AWS IoT konsol. PilihAWS IoT Greengrasskelompok, pilihPerangkat klientab, pilihGG_TrafficLight, pilihlahBayangan Perangkattab, dan kemudian pilihBayangan klasik.

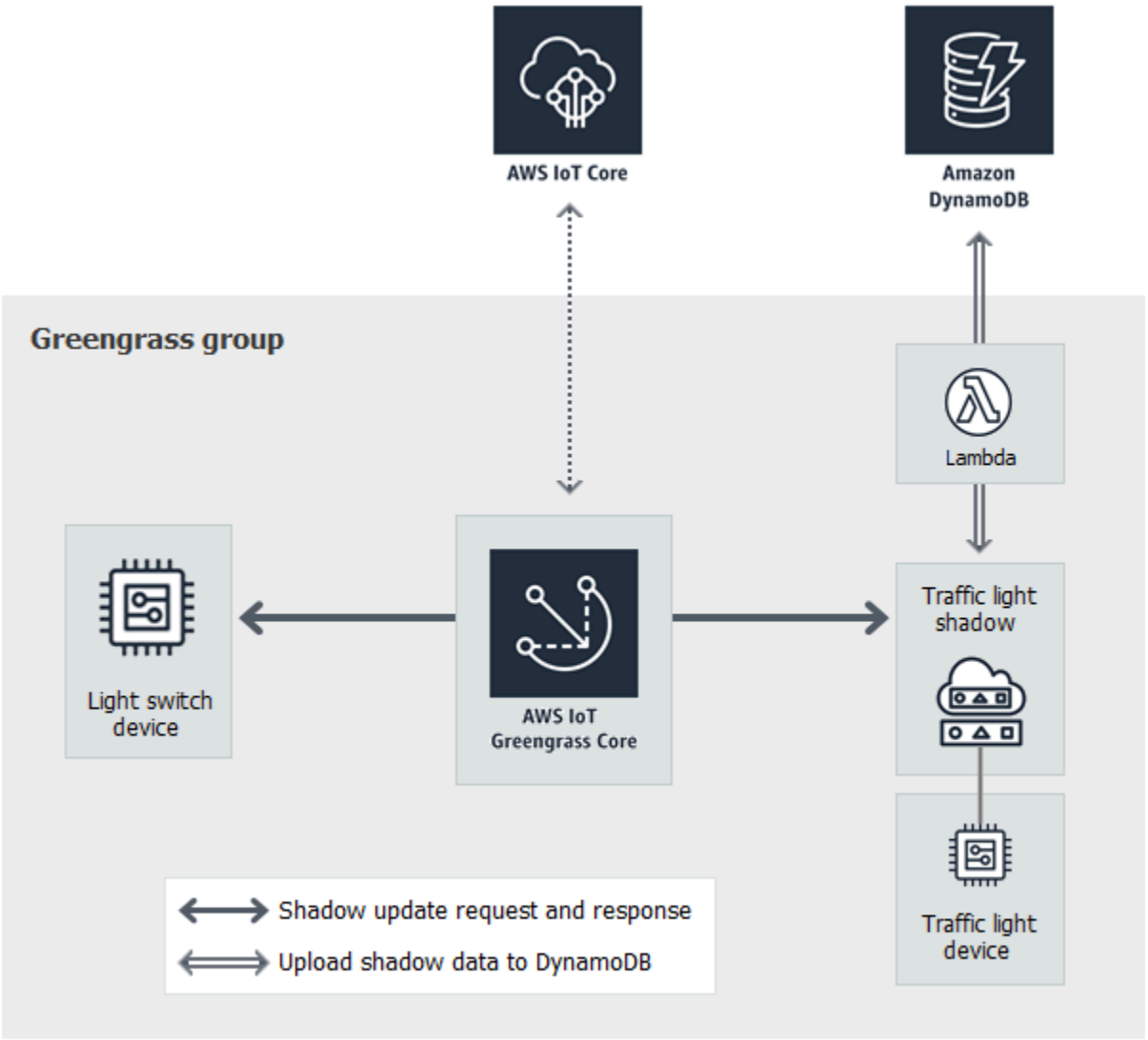
   Karena Anda mengaktifkan sinkronisasi GG_TrafficLight Bayangan untukAWS IoT, keadaan bayangan di cloud harus diperbarui setiap kali GGGG_Switch mengirimkan pembaruan. Fungsi ini dapat digunakan untuk mengekspos keadaan perangkat klien untukAWS IoT.

> **Note**
>
> Jika perlu, Anda dapat memecahkan masalah dengan melihat AWS IoT Greengrass catatan core, khususnya `runtime.log`:
>
> ```
> cd /greengrass/ggc/var/log
> sudo cat system/runtime.log | more
> ```
>
> Anda juga dapat melihat `GGShadowSyncManager.log` dan `GGShadowService.log`. Untuk informasi selengkapnya, lihat Memecahkan masalah.

Atur perangkat klien dan langganan klien. Anda menggunakannya di modul selanjutnya. Anda juga menjalankan perintah yang sama.

Modul 6: Mengakses lainnyaAWSjasa

Modul lanjutan ini menunjukkan cara AWS IoT Greengrass core dapat berinteraksi dengan AWS layanan di cloud. Ini dibangun di atas contoh lampu lalu lintas dari Modul 5 dan menambahkan fungsi Lambda yang memproses keadaan bayangan dan mengunggah ringkasan ke tabel Amazon DynamoDB.



Sebelum Anda memulai, jalankan skrip Pengaturan perangkat Greengrass ini, atau pastikan bahwa Anda telah menyelesaikan Modul 1 dan Modul 2. Anda juga harus menyelesaikan Modul 5. Anda tidak memerlukan komponen atau perangkat lain.

Modul ini akan memakan waktu sekitar 30 menit untuk menyelesaikannya.

> **Note**
>
> Modul ini membuat dan memperbarui tabel di DynamoDB. Meskipun sebagian besar operasi kecil dan termasuk dalam Amazon Web Services tingkat gratis, melakukan beberapa langkah

dalam modul ini mungkin mengakibatkan biaya ke akun Anda. Untuk informasi tentang harga, lihat dokumentasi Harga DynamoDB.

Topik

- Mengonfigurasi peran grup
- Buat fungsi Lambda dengan konsol
- Konfigurasi langganan
- Uji komunikasi

Mengonfigurasi peran grup

Peran grup adalah IAM role yang Anda buat dan lampirkan ke grup Greengrass Anda. Peran ini berisi izin yang men-deploy fungsi Lambda (dan fitur AWS IoT Greengrass lainnya) digunakan untuk mengakses AWS layanan. Untuk informasi selengkapnya, lihat the section called “Peran grup Greengrass”.

Anda menggunakan langkah-langkah tingkat tinggi berikut untuk membuat peran grup di konsol IAM.

1. Buat kebijakan yang mengizinkan atau menolak tindakan pada satu atau lebih sumber daya.
2. Buat peran yang menggunakan layanan Greengrass sebagai entitas terpercaya.
3. Lampirkan kebijakan Anda ke peran tersebut.

Kemudian, di AWS IoT konsol, Anda menambahkan peran ke grup Greengrass.

> **Note**
>
> Sebuah grup Greengrass memiliki satu peran grup. Jika Anda ingin menambahkan izin, Anda dapat mengedit kebijakan terlampir atau melampirkan kebijakan lainnya.

Untuk tutorial ini, Anda membuat kebijakan izin yang mengizinkan menjelaskan, membuat, dan memperbarui tindakan pada tabel Amazon DynamoDB. Kemudian, Anda melampirkan kebijakan untuk peran baru dan mengasosiasikan peran dengan grup Greengrass Anda.

Pertama, buat kebijakan yang dikelola pelanggan yang memberikan izin yang diperlukan oleh fungsi Lambda dalam modul ini.

1. Dalam kolom IAM, dalam panel navigas, pilih Kebijakan, lalu pilih Buat kebijakan.
2. Pada tab JSON ini, ganti konten placeholder dengan kebijakan berikut. Fungsi Lambda dalam modul ini menggunakan izin ini untuk membuat dan memperbarui tabel DynamoDB bernama `CarStats`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PermissionsForModule6",
            "Effect": "Allow",
            "Action": [
                "dynamodb:DescribeTable",
                "dynamodb:CreateTable",
                "dynamodb:PutItem"
            ],
            "Resource": "arn:aws:dynamodb:*:*:table/CarStats"
        }
    ]
}
```

3. Pilih Berikutnya: Tag, lalu pilih Berikutnya: Peninjauan. Tag tidak digunakan di tutorial ini.
4. Untuk Nama, masukkan **`greengrass_CarStats_Table`**, lalu pilih Buat kebijakan.

   Selanjutnya, buat peran yang menggunakan kebijakan baru.
5. Di panel navigasi, pilih Peran, lalu pilih Buat peran.
6. Di bawahTipe entitas tepercaya, pilihAWSlayanan.
7. Di bawahKasus penggunaan,Kasus penggunaan untuk kasus lainAWSjasapilihGreengrass, pilihGreengrass, dan kemudian pilihSelanjutnya.
8. Di bawahKebijakan izin, pilih yang baru**`greengrass_CarStats_Table`**kebijakan, kemudian pilihSelanjutnya.
9. Untuk Nama peran, masukkan **`Greengrass_Group_Role`**.
10. Untuk Description (Deskripsi), masukkan **`Greengrass group role for connectors and user-defined Lambda functions`**.

11. Pilih Create role (Buat peran).

    Sekarang, tambahkan peran ke grup Greengrass Anda.

12. DiAWS IoTpanel navigasi konsol, di bawahKelola, PerluasPerangkat Greengrass, dan kemudian pilihGrup (V1).
13. Di bawah grup Greengrass, pilih grup Anda.
14. MemiilihPengaturan, dan kemudian pilihPeran asosiasi.
15. MemiilihGreengrass_Group_Roledari daftar peran Anda, kemudian pilihPeran asosiasi.

Buat fungsi Lambda dengan konsol

Dalam langkah ini, Anda membuat fungsi Lambda yang melacak jumlah mobil yang melewati lampu lalu lintas. Setiap kali bahwa `GG_TrafficLight` keadaan bayangan mengubah ke `G`, fungsi Lambda mensimulasikan berlalunya sejumlah mobil acak (dari 1 sampai 20). Pada setiap perubahan cahaya `G` ketiga, fungsi Lambda mengirimkan statistik dasar, seperti min dan max, ke meja DynamoDB.

1. Di komputer Anda, buat folder dengan nama `car_aggregator`.
2. Dari folder TrafficLight contoh aktif GitHub, unduh`carAggregator.py` file ke`car_aggregator` folder. Ini kode fungsi Lambda Anda.

   > **Note**
   >
   > Contoh file Python ini disimpan dalam AWS IoT Greengrass repositori Core SDK untuk kenyamanan, tetapi tidak menggunakan AWS IoT Greengrass Core SDK.

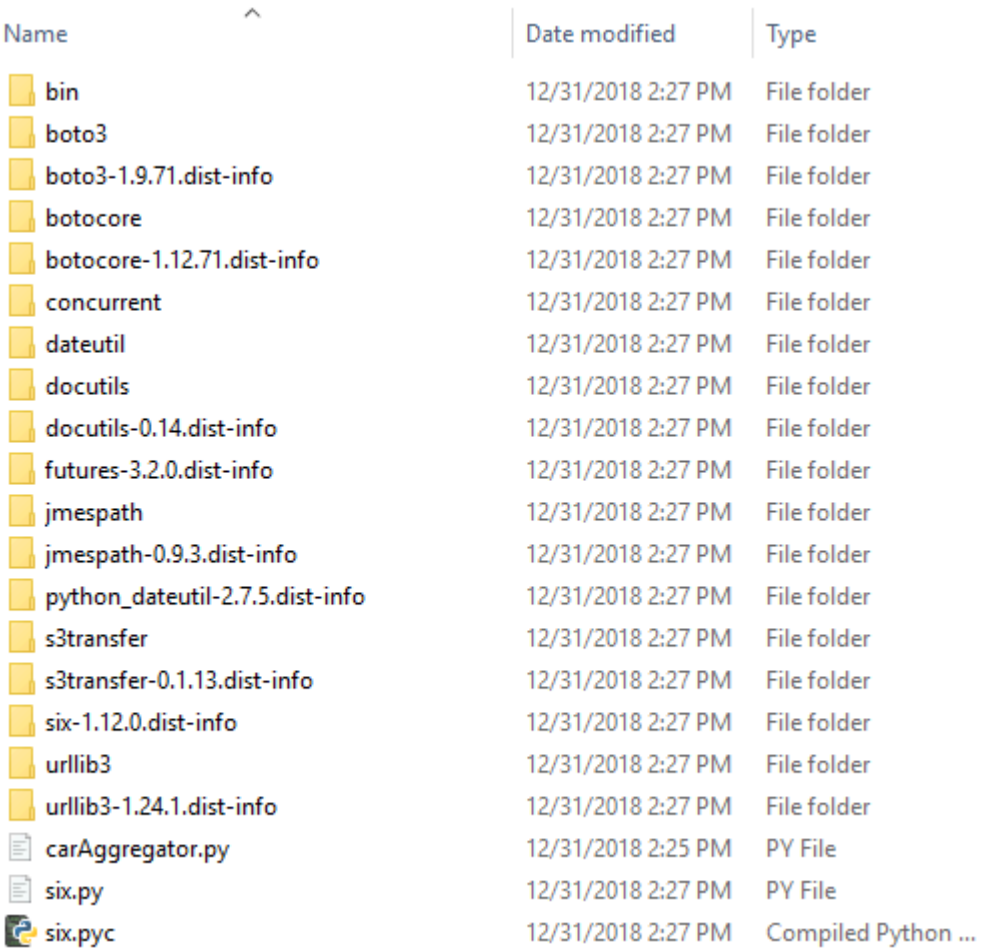
3. Jika Anda tidak bekerja di Wilayah Timur AS (N. Virgina), buka `carAggregator.py` dan ubah `region_name` di baris berikut ke Wilayah AWS yang sedang dipilih di konsol AWS IoT tersebut. Untuk daftarWilayah AWS s didukung, lihat AWS IoT Greengrassdi Referensi Umum Amazon Web Services.

   ```
   dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
   ```

4. Jalankan perintah berikut di jendela baris perintah untuk menginstal AWS SDK for Python (Boto3) dan dependensinya dalam `car_aggregator` folder. Fungsi Greengrass Lambda menggunakan AWS SDK untuk mengakses layanan AWS lainnya. (Untuk Windows, gunakan sebuah prompt perintah ditinggikan.)

```
pip install boto3 -t path-to-car_aggregator-folder
```

Hal ini menyebabkan daftar direktori yang serupa dengan berikut ini:

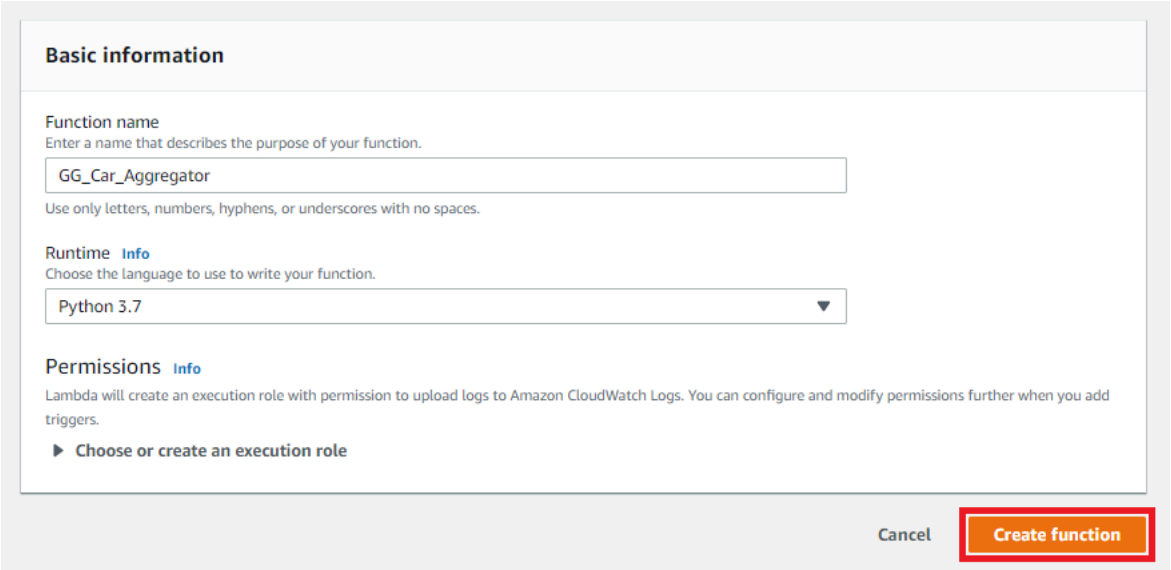


Name	Date modified	Type
bin	12/31/2018 2:27 PM	File folder
boto3	12/31/2018 2:27 PM	File folder
boto3-1.9.71.dist-info	12/31/2018 2:27 PM	File folder
botocore	12/31/2018 2:27 PM	File folder
botocore-1.12.71.dist-info	12/31/2018 2:27 PM	File folder
concurrent	12/31/2018 2:27 PM	File folder
dateutil	12/31/2018 2:27 PM	File folder
docutils	12/31/2018 2:27 PM	File folder
docutils-0.14.dist-info	12/31/2018 2:27 PM	File folder
futures-3.2.0.dist-info	12/31/2018 2:27 PM	File folder
jmespath	12/31/2018 2:27 PM	File folder
jmespath-0.9.3.dist-info	12/31/2018 2:27 PM	File folder
python_dateutil-2.7.5.dist-info	12/31/2018 2:27 PM	File folder
s3transfer	12/31/2018 2:27 PM	File folder
s3transfer-0.1.13.dist-info	12/31/2018 2:27 PM	File folder
six-1.12.0.dist-info	12/31/2018 2:27 PM	File folder
urllib3	12/31/2018 2:27 PM	File folder
urllib3-1.24.1.dist-info	12/31/2018 2:27 PM	File folder
carAggregator.py	12/31/2018 2:25 PM	PY File
six.py	12/31/2018 2:27 PM	PY File
six.pyc	12/31/2018 2:27 PM	Compiled Python ...

5. Kompres isi `car_aggregator` folder ke dalam folder `.zip` file bernama `car_aggregator.zip`. (Kompres isi folder, bukan folder.) Ini adalah paket deployment fungsi Lambda Anda.
6. Dalam konsol Lambda, buat fungsi bernama **`GG_Car_Aggregator`**, dan atur bidang yang tersisa sebagai berikut:

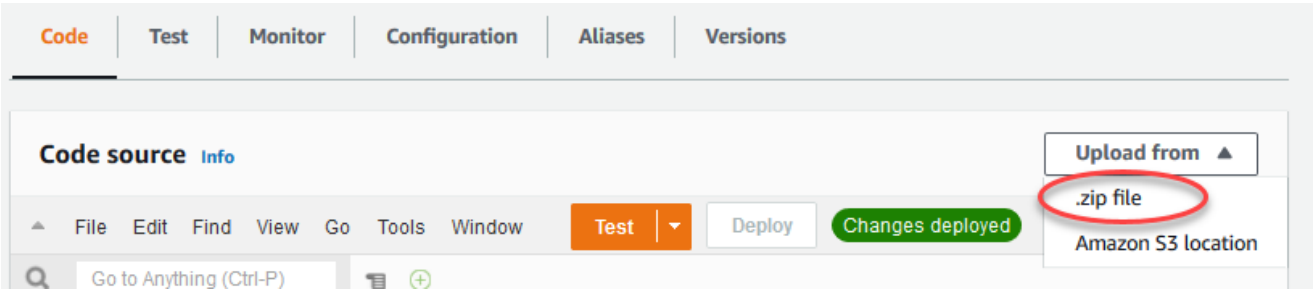
   - Untuk Waktu aktif, pilih Python 3.7.
   - Untuk Izin, pertahankan pengaturan default. Hal ini menciptakan peran eksekusi yang memberikan izin Lambda basic. Peran ini tidak digunakan oleh AWS IoT Greengrass.

   Pilih Buat fungsi.



7. Unggah paket deployment fungsi Lambda Anda:

   a. Pada tab Kode ini, di bawah Sumber kode, pilih Unggah dari. Dari dropdown, pilih file .zip.



   b. Pilih Unggah, lalu pilih paket deployment `car_aggregator.zip` Anda. Lalu, pilih Simpan.

   c. Pada tab Kode fungsi, di bawah Pengaturan waktu aktif, pilih Edit, dan kemudian masukkan nilai-nilai berikut.

      - Untuk Waktu pengoperasian, pilih Python 3.7.
      - Untuk Handler, masukkan **`carAggregator.function_handler`**

   d. Pilih Save (Simpan).

8. Terbitkan fungsi Lambda, lalu membuat alias bernama **`GG_CarAggregator`**. Untuk step-by-step petunjuk, lihat langkah-langkah untuk menerbitkan fungsi Lambda dan buat alias dalam Modul 3 (Bagian 1).

9. Di AWS IoT konsol, tambahkan fungsi Lambda yang baru saja Anda buat untuk grup AWS IoT Greengrass Anda:

a. Pada halaman konfigurasi grup, pilih fungsi Lambda, lalu di bawah fungsi Lambda Saya, pilih Tambah.

b. Untuk fungsi Lambda, pilih GG_Car_Aggregator.

c. Untuk versi fungsi Lambda, pilih alias untuk versi yang Anda terbitkan.

d. Untuk Batas memori, masukkan **`64 MB`**.

e. Untuk Disematkan, pilih Benar.

f. Pilih Tambahkan fungsi Lambda.

> **Note**
>
> Anda dapat menghapus fungsi Lambda lainnya dari modul sebelumnya.

Konfigurasi langganan

Pada langkah ini, Anda membuat langganan yang mengaktifkan G_TrafficLight bayangan untuk mengirim informasi keadaan yang diperbarui untuk fungsi G_Car_Aggregator Lambda. Langganan ini ditambahkan ke langganan yang Anda buat di Modul 5, yang semuanya diperlukan untuk modul ini.

1. Pada halaman konfigurasi grup, pilihLangganantab, dan kemudian pilihTambahkan.
2. PadaBuat langgananhalaman, lakukan hal berikut:

   a. UntukJenis Sumber, pilihLayanan, dan kemudian pilihLayanan Bayangan Lokal.

   b. UntukJenis target, pilihFungsi Lambda, dan kemudian pilihGG_Car_Aggregator.

   c. Untuk Filter topik, masukkan **`$aws/things/GG_TrafficLight/shadow/update/documents`**

   d. Pilih Buat langganan.

   Modul ini memerlukan langganan baru dan langganan yang Anda buat di Modul 5.
3. Pastikan bahwa Greengrass daemon berjalan, seperti yang dijelaskan dalam Men-deploy konfigurasi cloud ke perangkat core.
4. Pada halaman konfigurasi grup, pilihDeploy.

Uji komunikasi

1. Di komputer Anda, buka dua jendela baris perintah. Sama seperti diModul 5, satu jendela adalah untuk perangkat klien G_Switch dan yang lain adalah untuk G_TrafficLight perangkat klien. Anda menggunakannya untuk menjalankan perintah yang sama yang Anda jalankan di modul 5.

   Jalankan perintah berikut untuk perangkat klien G_Switch:

   ```
   cd path-to-certs-folder
   python lightController.py --endpoint AWS_IOT_ENDPOINT --rootCA AmazonRootCA1.pem
    --cert switchCertId-certificate.pem.crt --key switchCertId-private.pem.key --
   thingName GG_TrafficLight --clientId GG_Switch
   ```

   Jalankan perintah berikut untuk G_TrafficLight perangkat klien:

   ```
   cd path-to-certs-folder
   python trafficLight.py --endpoint AWS_IOT_ENDPOINT --rootCA AmazonRootCA1.pem --
   cert lightCertId-certificate.pem.crt --key lightCertId-private.pem.key --thingName
    GG_TrafficLight --clientId GG_TrafficLight
   ```

   Setiap 20 detik, switch memperbarui keadaan bayangan untuk G, Y, dan R, dan lampu menampilkan keadaan baru.

2. Fungsi handler dari fungsi Lambda dipicu pada setiap lampu hijau ketiga (setiap tiga menit), dan catatan DynamoDB baru dibuat. Setelah `lightController.py` dan `trafficLight.py` telah berjalan selama tiga menit, pergi ke AWS Management Console, dan buka konsol DynamoDB.
3. Pilih US East (N. Virginia) di Wilayah AWS menu. Ini adalah Wilayah di mana `GG_Car_Aggregator` fungsi membuat tabel.
4. Di panel navigasi, pilihTabel, dan kemudian pilihCarStatstabel.
5. PilihLihat itemuntuk melihat entri dalam tabel.

   Anda akan melihat entri dengan statistik dasar pada mobil berlalu (satu entri untuk setiap tiga menit). Anda mungkin butuh memilih tombol refresh untuk melihat pembaruan pada tabel.

6. Jika uji tidak berhasil, Anda dapat mencari informasi pemecahan masalah dalam catatan Greengrass.

   a. Beralih ke pengguna root dan arahkan ke `log` direktori. Akses ke AWS IoT Greengrass catatan memerlukan izin root.

```
sudo su
cd /greengrass/ggc/var/log
```

b. Periksa `runtime.log` untuk kesalahan.

```
cat system/runtime.log | grep 'ERROR'
```

c. Periksa log yang dihasilkan oleh fungsi Lambda.

```
cat user/region/account-id/GG_Car_Aggregator.log
```

Skrip `lightController.py` dan `trafficLight.py` menyimpan informasi koneksi di `groupCA` folder, yang dibuat dalam folder yang sama sebagai skrip. Jika Anda menerima eror koneksi, pastikan bahwa alamat IP di file `ggc-host` cocok dengan titik akhir alamat IP untuk core Anda.

Untuk informasi selengkapnya, lihat Memecahkan masalah.

Ini adalah akhir dari tutorial dasar. Anda sekarang harus memahamiAWS IoT Greengrassmodel pemrograman dan konsep dasarnya, termasukAWS IoT Greengrasscore, grup, langganan, perangkat klien, dan proses deployment untuk fungsi Lambda yang berjalan di edge.

Anda dapat menghapus tabel DynamoDB dan fungsi Greengrass Lambda dan langganan. Untuk menghentikan komunikasi antara AWS IoT Greengrass perangkat core dan AWS IoT cloud, buka terminal di perangkat core dan jalankan salah satu perintah berikut:

- Untuk mematikan AWS IoT Greengrass perangkat core:

```
sudo halt
```

- Untuk menghentikan AWS IoT Greengrass daemon:

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

Modul 7: Mensimulasikan integrasi keamanan perangkat keras

Fitur ini tersedia untuk AWS IoT Greengrass Core v1.7 dan yang lebih baru.

Modul lanjutan ini menunjukkan Anda cara mengonfigurasi modul keamanan perangkat keras yang disimulasikan (HSM) untuk digunakan dengan core Greengrass. Konfigurasi menggunakan SoftHSM, yang merupakan implementasi perangkat lunak murni yang menggunakan PKCS #11 antarmuka pemrograman aplikasi (API). Tujuan dari modul ini adalah untuk mengizinkan Anda untuk mengatur lingkungan di mana Anda dapat belajar dan melakukan pengujian awal terhadap implementasi perangkat lunak saja dari PKCS#11 API. Ini disediakan hanya untuk pembelajaran dan pengujian awal, bukan untuk penggunaan produksi dalam bentuk apapun.

Anda dapat menggunakan konfigurasi ini untuk bereksperimen dengan menggunakan layanan kompatibel-PKCS #11 untuk menyimpan kunci privat Anda. Untuk informasi lebih lanjut tentang implementasi perangkat lunak saja, lihat SoftHSM. Untuk informasi lebih lanjut tentang mengintegrasikan keamanan perangkat keras pada AWS IoT Greengrass core, termasuk persyaratan umum, lihat the section called “Integrasi keamanan perangkat keras”.

> **Important**
>
> Modul ini ditujukan untuk tujuan eksperimen saja. Kami sangat tidak menganjurkan penggunaan SoftHSM dalam lingkungan produksi karena mungkin menyediakan rasa keamanan tambahan yang salah. Konfigurasi yang dihasilkan tidak menyediakan manfaat keamanan yang sebenarnya. Kunci yang disimpan di SoftHSM tidak disimpan lebih aman daripada sarana penyimpanan rahasia lainnya di lingkungan Greengrass.
> Tujuan dari modul ini adalah untuk mengizinkan Anda untuk mempelajari tentang spesifikasi PKCS #11 dan melakukan pengujian awal perangkat lunak Anda jika Anda berencana untuk menggunakan HSM berbasis perangkat nyata di masa depan.
> Anda harus menguji implementasi perangkat keras masa depan Anda secara terpisah dan sepenuhnya sebelum penggunaan produksi karena mungkin ada perbedaan antara implementasi PKCS #11 yang disediakan di SoftHSM dan implementasi berbasis perangkat keras.

Jika Anda memerlukan bantuan terkait orientasi modul keamanan perangkat keras yang didukung, kontak AWS perwakilan Support Korporasi.

Sebelum Anda memulai, jalankan skrip Penyiapan Perangkat Greengrass ini, atau pastikan bahwa Anda telah menyelesaikan Modul 1 dan Modul 2 dari tutorial Memulai. Dalam modul ini, kami berasumsi bahwa core Anda sudah disediakan dan berkomunikasi dengan AWS. Modul ini akan memakan waktu sekitar 30 menit untuk menyelesaikannya.

Instal perangkat lunak SoftHSM

Dalam langkah ini, Anda memasang SoftHSM dan alat pkcs11, yang digunakan untuk mengelola keberlangsungan SoftHSM Anda.

- Di terminal pada AWS IoT Greengrass perangkat core Anda, jalankan perintah berikut:

  ```
  sudo apt-get install softhsm2 libsofthsm2-dev pkcs11-dump
  ```

  Untuk informasi lebih lanjut tentang paket ini, lihat Instal softhsm2, Pasang libsofthsm2-dev, dan Instal pkcs11-dump.

  > **Note**
  >
  > Jika Anda mengalami masalah ketika menggunakan perintah ini pada sistem Anda, lihat versi SoftHSM 2 di GitHub. Situs ini menyediakan informasi penginstalan lebih lanjut, termasuk cara membangun dari sumber.

Konfigurasi SoftHSM

Dalam langkah ini, Anda mengonfigurasi SoftHSM.

1. Beralih ke pengguna akar.

   ```
   sudo su
   ```

2. Gunakan halaman manual untuk menemukan `softhsm2.conf` lokasi yang menyeluruh dalam sistem. Lokasi yang umum adalah `/etc/softhsm/softhsm2.conf`, tetapi lokasi mungkin berbeda pada beberapa sistem.

   ```
   man softhsm2.conf
   ```

3. Buat direktori untuk file konfigurasi softhsm2 di lokasi yang menyeluruh dalam sistem. Dalam contoh ini, kami mengasumsikan lokasi `/etc/softhsm/softhsm2.conf`.

```
mkdir -p /etc/softhsm
```

4. Membuat direktori token di `/greengrass` direktori.

> **Note**
>
> Jika langkah ini dilewati, softhsm2-util melaporkan `ERROR: Could not initialize the library`.

```
mkdir -p /greengrass/softhsm2/tokens
```

5. Konfigurasikan direktori token.

```
echo "directories.tokendir = /greengrass/softhsm2/tokens" > /etc/softhsm/
softhsm2.conf
```

6. Konfigurasikan backend berbasis file.

```
echo "objectstore.backend = file" >> /etc/softhsm/softhsm2.conf
```

> **Note**
>
> Pengaturan konfigurasi ini ditujukan untuk tujuan eksperimen saja. Untuk melihat semua opsi konfigurasi, baca halaman manual buku panduan untuk file konfigurasi.
>
> ```
> man softhsm2.conf
> ```

Mengimpor kunci privat ke SoftHSM

Dalam langkah ini, Anda menginisialisasi token SoftHSM, mengkonversi format kunci privat, lalu mengimpor kunci privat.

1. Menginisialisasi token SoftHSM.

```
softhsm2-util --init-token --slot 0 --label greengrass --so-pin 12345 --pin 1234
```

> **Note**
>
> Jika diminta, masukkan pin SO dari `12345` dan pin pengguna dari `1234`. AWS IoT Greengrass tidak menggunakan pin SO (supervisor), sehingga Anda dapat menggunakan nilai apapun.
> Jika Anda menerima kesalahan `CKR_SLOT_ID_INVALID: Slot 0 does not exist`, coba perintah berikut:
>
> ```
> softhsm2-util --init-token --free --label greengrass --so-pin 12345 --pin
>  1234
> ```

2. Konversikan kunci privat ke format yang dapat digunakan oleh alat impor SoftHSM. Untuk tutorial ini, Anda mengubah kunci privat yang Anda peroleh dari opsi pembuatan Grup Default di dalam Modul 2 dari tutorial Memulai.

   ```
   openssl pkcs8 -topk8 -inform PEM -outform PEM -nocrypt -in hash.private.key -
   out hash.private.pem
   ```

3. Impor kunci privat ke SoftHSM. Jalankan hanya salah satu perintah berikut, tergantung pada versi softhsm2-util Anda.

   Sintaks Raspbian softhsm2-util v2.2.0

   ```
   softhsm2-util --import hash.private.pem --token greengrass --label iotkey --id
    0000 --pin 12340
   ```

   Sintaks Ubuntu softhsm2-util v2.0.0

   ```
   softhsm2-util --import hash.private.pem --slot 0 --label iotkey --id 0000 --pin
    1234
   ```

   Perintah ini mengidentifikasi slot sebagai `0` dan mendefinisikan label kunci sebagai `iotkey`. Anda menggunakan nilai-nilai ini di bagian selanjutnya.

Setelah kunci privat diimpor, Anda dapat secara opsional menghapusnya dari direktori `/greengrass/certs` ini. Pastikan untuk menyimpan CA akar dan sertifikat perangkat dalam direktori.

Konfigurasikan core Greengrass untuk menggunakan SoftHSM

Dalam langkah ini, Anda memodifikasi file konfigurasi core Greengrass untuk menggunakan SoftHSM.

1. Temukan path ke pustaka penyedia SoftHSM (`libsofthsm2.so`) pada sistem Anda:

   a. Dapatkan daftar paket yang diinstal untuk perpustakaan.

   ```
   sudo dpkg -L libsofthsm2
   ```

   File `libsofthsm2.so` terletak di `softhsm` direktori.

   b. Salin jalur lengkap ke file (sebagai contoh, `/usr/lib/x86_64-linux-gnu/softhsm/libsofthsm2.so`). Anda menggunakan nilai ini kemudian.

2. Hentikan daemon Greengrass.

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

3. Buka file konfigurasi Greengrass. Ini adalah file `config.json` dalam direktori `/greengrass/config` ini.

   > **Note**
   >
   > Contoh dalam prosedur ini ditulis dengan asumsi bahwa file `config.json` menggunakan format yang dihasilkan dari opsi Pembuatan Grup default di dalam Modul 2 dari tutorial Memulai.

4. Di dalam `crypto.principals` objek, masukkan objek sertifikat server MQTT berikut. Tambahkan koma di mana diperlukan untuk membuat file JSON valid.

   ```
   "MQTTServerCertificate": {
     "privateKeyPath": "path-to-private-key"
   }
   ```

5. Di `crypto` objek, masukkan `PKCS11` objek berikut. Tambahkan koma di mana diperlukan untuk membuat file JSON valid.

```
  "PKCS11": {
    "P11Provider": "/path-to-pkcs11-provider-so",
    "slotLabel": "crypto-token-name",
    "slotUserPin": "crypto-token-user-pin"
  }
```

File Anda akan terlihat serupa dengan berikut ini:

```
{
  "coreThing" : {
    "caPath" : "root.ca.pem",
    "certPath" : "hash.cert.pem",
    "keyPath" : "hash.private.key",
    "thingArn" : "arn:partition:iot:region:account-id:thing/core-thing-name",
    "iotHost" : "host-prefix.iot.region.amazonaws.com",
    "ggHost" : "greengrass.iot.region.amazonaws.com",
    "keepAlive" : 600
  },
  "runtime" : {
    "cgroup" : {
      "useSystemd" : "yes"
    }
  },
  "managedRespawn" : false,
  "crypto": {
    "PKCS11": {
      "P11Provider": "/path-to-pkcs11-provider-so",
      "slotLabel": "crypto-token-name",
      "slotUserPin": "crypto-token-user-pin"
    },
    "principals" : {
      "MQTTServerCertificate": {
        "privateKeyPath": "path-to-private-key"
      },
      "IoTCertificate" : {
        "privateKeyPath" : "file:///greengrass/certs/hash.private.key",
        "certificatePath" : "file:///greengrass/certs/hash.cert.pem"
      },
      "SecretsManager" : {
```

```
        "privateKeyPath" : "file:///greengrass/certs/hash.private.key"
      }
    },
    "caPath" : "file:///greengrass/certs/root.ca.pem"
  }
}
```

> **Note**
>
> Untuk menggunakan over-the-air (OTA) update dengan keamanan perangkat keras,`PKCS11`objek juga harus berisi`OpenSSLEngine`properti Untuk informasi selengkapnya, lihat the section called “Konfigurasi pembaruan OTA”.

6. Edit `crypto` objek:

   a. Konfigurasikan `PKCS11` objek.

      - Untuk `P11Provider`, masukkan path lengkap ke `libsofthsm2.so`.
      - Untuk `slotLabel`, masukkan `greengrass`.
      - Untuk `slotUserPin`, masukkan `1234`.

   b. Konfigurasikan path kunci privat di dalam `principals` objek. Jangan edit `certificatePath` properti.

      - Untuk `privateKeyPath` properti, masukkan path RFC 7512 PKCS #11 berikut (yang menentukan label kunci). Lakukan ini untuk `IoTCertificate`, `SecretsManager`, dan `MQTTServerCertificate` pelaku utama.

        ```
        pkcs11:object=iotkey;type=private
        ```

   c. Periksa `crypto` objek. Itu akan terlihat serupa dengan yang berikut ini:

      ```
      "crypto": {
        "PKCS11": {
          "P11Provider": "/usr/lib/x86_64-linux-gnu/softhsm/libsofthsm2.so",
          "slotLabel": "greengrass",
          "slotUserPin": "1234"
        },
        "principals": {
          "MQTTServerCertificate": {
            "privateKeyPath": "pkcs11:object=iotkey;type=private"
      ```

```
      },
      "SecretsManager": {
        "privateKeyPath": "pkcs11:object=iotkey;type=private"
      },
      "IoTCertificate": {
        "certificatePath": "file://certs/core.crt",
        "privateKeyPath": "pkcs11:object=iotkey;type=private"
      }
    },
    "caPath": "file://certs/root.ca.pem"
  }
```

7. Hapus nilai `caPath`, `certPath`, dan `keyPath` dari `coreThing` objek. Itu akan terlihat serupa dengan yang berikut ini:

```
"coreThing" : {
  "thingArn" : "arn:partition:iot:region:account-id:thing/core-thing-name",
  "iotHost" : "host-prefix-ats.iot.region.amazonaws.com",
  "ggHost" : "greengrass-ats.iot.region.amazonaws.com",
  "keepAlive" : 600
}
```

> **Note**
>
> Untuk tutorial ini, Anda menentukan kunci privat yang sama untuk semua pelaku utama. Untuk informasi lebih lanjut tentang memilih kunci privat untuk server MQTT lokal, lihat Performa. Untuk informasi lebih lanjut tentang secrets manager lokal, lihat Men-deploy rahasia ke core.

Uji konfigurasi

- Mulai daemon Greengrass.

```
cd /greengrass/ggc/core/
sudo ./greengrassd start
```

Jika daemon dimulai dengan sukses, maka core Anda telah dikonfigurasi dengan benar.

Anda sekarang siap untuk mempelajari spesifikasi PKCS #11 dan melakukan pengujian awal dengan API PKCS#11 yang disediakan oleh implementasi SoftHSM.

> **Important**
>
> Sekali lagi, sangat penting untuk menyadari bahwa modul ini ditujukan untuk pembelajaran dan pengujian saja. Ini tidak benar-benar meningkatkan postur keamanan lingkungan Greengrass Anda.
> Sebaliknya, tujuan dari modul ini adalah untuk mengizinkan Anda untuk mulai pembelajaran dan pengujian dalam persiapan untuk menggunakan HSM berbasis hardware nyata di masa depan. Pada saat itu, Anda harus secara terpisah dan sepenuhnya menguji perangkat lunak Anda terhadap HSM berbasis perangkat keras sebelum penggunaan produksi, karena mungkin ada perbedaan antara implementasi PKCS #11 yang disediakan di SoftHSM dan implementasi berbasis perangkat keras.

Lihat juga

- Panduan Penggunaan Antarmuka Token Kriptografi PKCS #11 Versi 2.40. Diedit oleh John Leiseboer dan Robert Griffin. 16 November 2014. Catatan Komite OASIS 02. http://docs.oasis-open.org/pkcs11/pkcs11-ug/v2.40/cn02/pkcs11-ug-v2.40-cn02.html. Versi terbaru: http://docs.oasis-open.org/pkcs11/pkcs11-ug/v2.40/pkcs11-ug-v2.40.html.
- RFC 7512

Perbarui OTA AWS IoT Greengrass perangkat lunak Core

Paket perangkat lunak AWS IoT Greengrass Core mencakup agen pembaruan yang dapat melakukan pembaruan AWS IoT Greengrass perangkat lunak over-the-air (OTA). Anda dapat menggunakan pembaruan OTA untuk menginstal versi terbaru AWS IoT Greengrass perangkat lunak Core atau perangkat lunak agen pembaruan OTA pada satu atau lebih core. Dengan pembaruan OTA, perangkat inti Anda tidak harus hadir secara fisik.

Kami merekomendasikan agar Anda menggunakan pembaruan OTA bila memungkinkan. Mereka menyediakan mekanisme yang dapat Anda gunakan untuk melacak status pembaruan dan sejarah pembaruan. Jika terjadi gagal pembaruan, agen pembaruan OTA akan kembali ke versi perangkat lunak sebelumnya.

> **Note**
>
> pembaruan OTA tidak didukung saat Anda menggunakan `apt` untuk menginstal AWS IoT Greengrass perangkat lunak Core. Untuk instalasi ini, kami sarankan Anda menggunakan `apt` untuk meng-upgrade perangkat lunak. Untuk informasi selengkapnya, lihat the section called “Instal dari repositori APT”.

pembaruan OTA membuatnya lebih efisien untuk:

- Memperbaiki kerentanan keamanan.
- Mengatasi masalah stabilitas perangkat lunak.
- Men-deploy fitur baru atau yang lebih baik.

Fitur ini terintegrasi dengan AWS IoT tugas.

Persyaratan

Persyaratan berikut berlaku untuk pembaruan OTA AWS IoT Greengrass perangkat lunak.

- Greengrass core harus memiliki setidaknya 400 MB ruang disk yang tersedia di penyimpanan lokal. Agen pembaruan OTA memerlukan sekitar tiga kali persyaratan penggunaan waktu aktif AWS IoT Greengrass perangkat lunak Core. Untuk informasi selengkapnya, lihat Kuota layanan untuk inti Greengrass di halaman. Referensi Umum Amazon Web Services

- Greengrass core harus memiliki koneksi ke AWS Cloud.
- Greengrass core harus dikonfigurasi dengan benar dan ditetapkan dengan sertifikat dan kunci untuk otentikasi dengan AWS IoT Core dan AWS IoT Greengrass. Untuk informasi selengkapnya, lihat the section called “Sertifikat X.509”.
- Greengrass core tidak dapat dikonfigurasi untuk menggunakan proksi jaringan.

> **Note**
>
> Dimulai di AWS IoT Greengrass v1.9.3, pembaruan OTA didukung pada core yang mengonfigurasi lalu lintas MQTT menggunakan port 443 bukannya port default 8883. Namun, agen pembaruan OTA tidak mendukung pembaruan melalui proksi jaringan. Untuk informasi selengkapnya, lihat the section called “Connect pada port 443 atau melalui proksi jaringan”.

- Boot terpercaya tidak dapat diaktifkan di partisi yang berisi AWS IoT Greengrass perangkat lunak Core.

> **Note**
>
> Anda dapat menginstal dan menjalankan AWS IoT Greengrass perangkat lunak Core pada partisi yang telah dipercaya boot diaktifkan, tetapi pembaruan OTA tidak didukung.

- AWS IoT Greengrass harus memiliki izin baca/tulis pada partisi yang berisi AWS IoT Greengrass perangkat lunak Core.
- Jika Anda menggunakan sistem init untuk mengelola Core Greengrass Anda, Anda harus mengonfigurasi pembaruan OTA untuk mengintegrasikan dengan sistem init. Untuk informasi selengkapnya, lihat the section called “Integrasi dengan sistem init”.
- Anda harus membuat peran yang digunakan untuk presign URL Amazon S3 ke artefak pembaruan perangkat lunak AWS IoT Greengrass ini. Peran signer ini mengizinkan AWS IoT Core untuk mengakses artefak pembaruan perangkat lunak yang disimpan di Amazon S3 atas nama Anda. Untuk informasi selengkapnya, lihat the section called “Izin IAM untuk pembaruan OTA”.

Izin IAM untuk pembaruan OTA

Ketika AWS IoT Greengrass merilis versi baru dari perangkat lunak Core AWS IoT Greengrass ini, AWS IoT Greengrass memperbarui artefak perangkat lunak yang disimpan di Amazon S3 yang digunakan untuk pembaruan OTA.

Anda Akun AWS harus menyertakan peran signer URL Amazon S3 yang dapat digunakan untuk mengakses artefak ini. Peran harus memiliki kebijakan izin yang mengizinkan `s3:GetObject` tindakan pada bucket di target Wilayah AWSs. Peran ini juga harus memiliki kebijakan kepercayaan yang mengizinkan `iot.amazonaws.com` untuk mengambil peran sebagai entitas terpercaya.

Kebijakan perizinan

Untuk izin peran, Anda dapat menggunakan kebijakan terkelola AWS atau buat kebijakan kustom.

- Menggunakan kebijakan AWS terkelola

  Kebijakan yang UpdateArtifactAccess dikelola GreenGrassota disediakan oleh. AWS IoT Greengrass Gunakan kebijakan ini jika Anda ingin mengizinkan akses di semua Amazon Web Services Region didukung oleh AWS IoT Greengrass, baik saat ini maupun masa depan.
- Membuat kebijakan kustom

  Anda harus membuat kebijakan kustom jika Anda ingin secara eksplisit menentukan Wilayah Amazon Web Services di mana core Anda di-deploy. Contoh kebijakan berikut mengizinkan akses ke pembaruan perangkat lunak AWS IoT Greengrass di enam Wilayah.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAccessToGreengrassOTAUpdateArtifacts",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::us-east-1-greengrass-updates/*",
                "arn:aws:s3:::us-west-2-greengrass-updates/*",
                "arn:aws:s3:::ap-northeast-1-greengrass-updates/*",
                "arn:aws:s3:::ap-southeast-2-greengrass-updates/*",
                "arn:aws:s3:::eu-central-1-greengrass-updates/*",
                "arn:aws:s3:::eu-west-1-greengrass-updates/*"
```

```
            ]
        }
    ]
}
```

Kebijakan kepercayaan

Kebijakan kepercayaan yang melekat pada peran harus mengizinkan `sts:AssumeRole` tindakan dan menentukan `iot.amazonaws.com` sebagai pelaku utama. Ini mengizinkan AWS IoT Core untuk menganggap peran sebagai entitas terpercaya. Berikut ini adalah contoh dokumen kebijakan:

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

Selain itu, pengguna yang memulai pembaruan OTA harus memiliki izin untuk menggunakan `greengrass:CreateSoftwareUpdateJob` dan `iot:CreateJob`, dan menggunakan `iam:PassRole` untuk lulus izin dari peran signer. Berikut ini adalah contoh kebijakan IAM:

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

```
            "Effect": "Allow",
            "Action": [
                "iot:CreateJob"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "arn-of-s3-url-signer-role"
        }
    ]
}
```

Pertimbangan-pertimbangan

Sebelum Anda meluncurkan pembaruan OTA perangkat lunak Core Greengrass, waspadai dampak pada perangkat dalam grup Greengrass Anda, baik pada perangkat core dan pada perangkat klien yang terhubung secara lokal ke core itu:

- Core menutup selama pembaruan.
- Setiap fungsi Lambda berjalan pada core dimatikan. Jika fungsi tersebut tulis ke sumber daya lokal, mereka mungkin meninggalkan sumber daya tersebut dalam keadaan salah kecuali dimatikan dengan benar.
- Selama downtime core, semua koneksinya dengan AWS Cloud hilang. Pesan yang diarahkan melalui core oleh perangkat klien hilang.
- Cache kredensia hilang.
- Antrian yang memegang pekerjaan tertunda untuk fungsi Lambda hilang.
- Fungsi Lambda yang berumur panjang kehilangan informasi status dinamis mereka dan semua pekerjaan yang tertunda dijatuhkan.

Informasi status berikut dipertahankan selama pembaruan OTA:

- Konfigurasi core
- Konfigurasi grup Greengrass

- Bayangan lokal
- Greengrass log
- Log agen pembaruan OTA

Agen pembaruan OTA Greengrass

Agen pembaruan OTA Greengrass adalah komponen perangkat lunak pada perangkat yang menangani pembaruan pekerjaan yang dibuat dan di-deploy di cloud. Agen pembaruan OTA didistribusikan dalam paket perangkat lunak yang sama sebagai AWS IoT Greengrass perangkat lunak Core. Agen tersebut berada di */greengrass-root*`/ota/ota_agent/ggc-ota`. Ini menulis log ke `/var/log/greengrass/ota/ggc_ota.txt`.

> **Note**
>
> */greengrass-root* mewakili jalan di mana AWS IoT Greengrass perangkat lunak Core diinstal pada perangkat Anda. Biasanya, ini adalah `/greengrass` direktori.

Anda dapat memulai agen pembaruan OTA dengan mengeksekusi biner secara manual atau dengan mengintegrasikannya sebagai bagian dari skrip init, seperti file layanan systemd. Jika Anda menjalankan biner secara manual, itu harus dijalankan sebagai root. Ketika ini mulai, agen pembaruan OTA mendengarkan tugas pembaruan perangkat lunak AWS IoT Greengrass dari AWS IoT Core dan mengeksekusi mereka secara berurutan. Agen pembaruan OTA mengabaikan semua lainnya AWS IoT jenis tugas.

Kutipan berikut menunjukkan contoh file layanan systemd untuk memulai, berhenti, dan restart agen pembaruan OTA:

```
[Unit]
Description=Greengrass OTA Daemon

[Service]
Type=forking
Restart=on-failure
ExecStart=/greengrass/ota/ota_agent/ggc-ota

[Install]
WantedBy=multi-user.target
```

Core yang merupakan target pembaruan tidak harus menjalankan dua instans agen pembaruan OTA. Melakukan hal itu membuat kedua agen untuk memproses tugas yang sama, yang membuat konflik.

Integrasi dengan sistem init

Selama pembaruan OTA, agen pembaruan OTA restart binari pada perangkat core. Jika bineri berjalan, ini dapat menyebabkan konflik ketika sistem init memantau keadaan perangkat lunak Core AWS IoT Greengrass atau agen selama pembaruan. Untuk membantu mengintegrasikan mekanisme pembaruan OTA dengan strategi pemantauan init Anda, Anda dapat menulis skrip shell yang berjalan sebelum dan sesudah pembaruan. Sebagai contoh, Anda dapat menggunakan `ggc_pre_update.sh` skrip untuk membuat cadangan data atau menghentikan proses sebelum perangkat dimatikan.

Untuk memberitahu agen pembaruan OTA untuk menjalankan skrip ini, Anda harus menyertakan `"managedRespawn" : true` bendera di file config.json. Pengaturan ini ditunjukkan dalam contoh berikut:

```
{
   "coreThing": {
       …
   },
   "runtime": {
       …
   },
   "managedRespawn": true
       …
}
```

Respawn terkelola dengan pembaruan OTA

Persyaratan berikut berlaku untuk pembaruan OTA dengan `managedRespawn` diatur ke `true`:

- Skrip shell berikut harus hadir di */greengrass-root*`/usr/scripts` direktori:
  - `ggc_pre_update.sh`
  - `ggc_post_update.sh`
  - `ota_pre_update.sh`
  - `ota_post_update.sh`

- Skrip harus mengembalikan kode kembali sukses.
- Skrip harus dimiliki oleh root dan executable oleh root saja.
- Skrip `ggc_pre_update.sh` harus menghentikan Greengrass daemon.
- Skrip `ggc_post_update.sh` harus mulai Greengrass daemon.

> **Note**
>
> Karena agen pembaruan OTA mengelola proses sendiri, `ota_pre_update.sh` dan `ota_post_update.sh` skrip tidak perlu menghentikan atau memulai layanan OTA.

Agen pembaruan OTA menjalankan skrip dari */greengrass-root*`/usr/scripts`. Pohon direktori akan terlihat seperti berikut:

```
<greengrass_root>
|-- certs
|-- config
|   |-- config.json
|-- ggc
|-- usr/scripts
|   |-- ggc_pre_update.sh
|   |-- ggc_post_update.sh
|   |-- ota_pre_update.sh
|   |-- ota_post_update.sh
|-- ota
```

Ketika `managedRespawn` diatur ke `true`, agen pembaruan OTA memeriksa direktori */greengrass-root*`/usr/scripts` untuk skrip ini sebelum dan sesudah pembaruan perangkat lunak. Jika skrip tidak ada, pembaruan gagal. AWS IoT Greengrass tidak memvalidasi isi skrip ini. Sebagai praktik terbaik, verifikasi bahwa skrip Anda berfungsi dengan benar dan mengeluarkan kode keluar yang sesuai untuk kesalahan.

Untuk pembaruan OTA dari perangkat lunak Core AWS IoT Greengrass ini:

- Sebelum memulai pembaruan, agen menjalankan `ggc_pre_update.sh` skrip. Gunakan skrip ini untuk perintah yang perlu dijalankan sebelum agen pembaruan OTA dimulai AWS IoT Greengrass updafe perangkat lunak core, seperti mencadangkan data atau menghentikan proses yang sedang berjalan. Contoh berikut menunjukkan skrip sederhana untuk menghentikan Greengrass daemon.

```
#!/bin/bash
set -euo pipefail
systemctl stop greengrass
```

- Setelah menyelesaikan pembaruan, agen menjalankan skrip `ggc_post_update.sh` ini. Gunakan skrip ini untuk perintah yang perlu dijalankan setelah agen pembaruan OTA memulai pembaruan perangkat lunak core AWS IoT Greengrass ini, seperti untuk me-restart proses. Contoh berikut menunjukkan skrip sederhana untuk memulai Greengrass daemon.

```
#!/bin/bash
set -euo pipefail
systemctl start greengrass
```

Untuk pembaruan OTA dari agen pembaruan OTA:

- Sebelum memulai pembaruan, agen menjalankan `ota_pre_update.sh` skrip. Gunakan skrip ini untuk perintah yang perlu dijalankan sebelum agen pembaruan OTA memperbarui sendiri, seperti untuk membuat cadangan data atau menghentikan proses yang berjalan.
- Setelah menyelesaikan pembaruan, agen menjalankan skrip `ota_post_update.sh` ini. Gunakan skrip ini untuk perintah yang perlu dijalankan setelah agen pembaruan OTA memperbarui sendiri, seperti untuk me-restart proses.

> **Note**
>
> Jika `managedRespawn` diatur ke `false`, agen pembaruan OTA tidak menjalankan skrip.

Buat pembaruan OTA

Ikuti langkah-langkah ini untuk melakukan pembaruan OTA dari perangkat lunak AWS IoT Greengrass pada satu atau lebih core:

1. Pastikan bahwa core Anda memenuhi persyaratan untuk pembaruan OTA.

> **Note**
>
> Jika Anda mengonfigurasi sistem init untuk mengelola perangkat lunak Core AWS IoT Greengrass atau agen pembaruan OTA, verifikasi hal berikut pada core Anda:
>
> - File config.json menentukan `"managedRespawn" : true`.
> - Direktori /*greengrass-root*/usr/scripts berisi skrip berikut:
>   - ggc_pre_update.sh
>   - ggc_post_update.sh
>   - ota_pre_update.sh
>   - ota_post_update.sh
>
> Untuk informasi selengkapnya, lihat the section called “Integrasi dengan sistem init”.

2. Di terminal perangkat core, mulai agen pembaruan OTA.

```
cd /greengrass-root/ota/ota_agent
sudo ./ggc-ota
```

> **Note**
>
> *greengrass-root* mewakili jalan di mana AWS IoT Greengrass perangkat lunak core diinstal pada perangkat Anda. Biasanya, ini adalah `/greengrass` direktori.

Jangan mulai beberapa contoh agen pembaruan OTA pada core karena dapat menyebabkan konflik.

3. Gunakan AWS IoT Greengrass API untuk membuat pekerjaan pembaruan perangkat lunak.

   a. Panggil CreateSoftwareUpdateJobAPI. Dalam contoh prosedur ini, kami menggunakan AWS CLI Perintah.

      Perintah berikut membuat tugas yang memperbarui perangkat lunak Core AWS IoT Greengrass pada satu core. Ganti nilai contoh dan kemudian jalankan perintah.

Linux or macOS terminal

```
aws greengrass create-software-update-job \
--update-targets-architecture x86_64 \
--update-targets [\"arn:aws:iot:region:123456789012:thing/myCoreDevice\"] \
--update-targets-operating-system ubuntu \
--software-to-update core \
--s3-url-signer-role arn:aws:iam::123456789012:role/myS3UrlSignerRole \
--update-agent-log-level WARN \
--amzn-client-token myClientToken1
```

Windows command prompt

```
aws greengrass create-software-update-job ^
--update-targets-architecture x86_64 ^
--update-targets [\"arn:aws:iot:region:123456789012:thing/myCoreDevice\"] ^
--update-targets-operating-system ubuntu ^
--software-to-update core ^
--s3-url-signer-role arn:aws:iam::123456789012:role/myS3UrlSignerRole ^
--update-agent-log-level WARN ^
--amzn-client-token myClientToken1
```

Perintah mengembalikan respons berikut.

```
{
    "IotJobId": "GreengrassUpdateJob_c3bd7f36-ee80-4d42-8321-a1da0EXAMPLE",
    "IotJobArn": "arn:aws:iot:region:123456789012:job/
GreengrassUpdateJob_c3bd7f36-ee80-4d42-8321-a1da0EXAMPLE",
    "PlatformSoftwareVersion": "1.10.1"
}
```

b. Salin `IoTJobId` dari respon.

c. Panggil DescribeJobAWS IoT CoreAPI untuk melihat status pekerjaan. Ganti nilai contoh dengan ID tugas Anda dan kemudian jalankan perintah.

```
aws iot describe-job --job-id GreengrassUpdateJob_c3bd7f36-ee80-4d42-8321-
a1da0EXAMPLE
```

Perintah mengembalikan objek respon yang berisi informasi tentang tugas, termasuk `status` dan `jobProcessDetails`.


```
{
    "job": {
        "jobArn": "arn:aws:iot:region:123456789012:job/
GreengrassUpdateJob_c3bd7f36-ee80-4d42-8321-a1da0EXAMPLE",
        "jobId": "GreengrassUpdateJob_c3bd7f36-ee80-4d42-8321-a1da0EXAMPLE",
        "targetSelection": "SNAPSHOT",
        "status": "IN_PROGRESS",
        "targets": [
            "arn:aws:iot:region:123456789012:thing/myCoreDevice"
        ],
        "description": "This job was created by Greengrass to update the
 Greengrass Cores in the targets with version 1.10.1 of the core software
 running on x86_64 architecture.",
        "presignedUrlConfig": {
            "roleArn": "arn:aws::iam::123456789012:role/myS3UrlSignerRole",
            "expiresInSec": 3600
        },
        "jobExecutionsRolloutConfig": {},
        "createdAt": 1588718249.079,
        "lastUpdatedAt": 1588718253.419,
        "jobProcessDetails": {
            "numberOfCanceledThings": 0,
            "numberOfSucceededThings": 0,
            "numberOfFailedThings": 0,
            "numberOfRejectedThings": 0,
            "numberOfQueuedThings": 1,
            "numberOfInProgressThings": 0,
            "numberOfRemovedThings": 0,
            "numberOfTimedOutThings": 0
        },
        "timeoutConfig": {}
    }
}
```


Untuk bantuan pemecahan masalah, lihat Memecahkan masalah.

API CreateSoftwareUpdateJob

Anda dapat menggunakan API `CreateSoftwareUpdateJob` untuk pembaruan perangkat lunak Core AWS IoT Greengrass atau perangkat lunak agen pembaruan OTA pada perangkat core Anda. API ini membuat tugas snapshot AWS IoT yang memberitahu perangkat ketika pembaruan tersedia. Setelah Anda memanggil `CreateSoftwareUpdateJob`, Anda dapat menggunakan perintah tugas AWS IoT lainnya untuk melacak pembaruan perangkat lunak. Untuk informasi selengkapnya, lihat Tugas di AWS IoT Panduan Developer.

Contoh berikut menunjukkan cara menggunakan AWS CLI untuk membuat tugas yang memperbarui perangkat lunak Core AWS IoT Greengrass pada perangkat core:

```
aws greengrass create-software-update-job \
--update-targets-architecture x86_64 \
--update-targets [\"arn:aws:iot:region:123456789012:thing/myCoreDevice\"] \
--update-targets-operating-system ubuntu \
--software-to-update core \
--s3-url-signer-role arn:aws:iam::123456789012:role/myS3UrlSignerRole \
--update-agent-log-level WARN \
--amzn-client-token myClientToken1
```

Perintah `create-software-update-job` mengembalikan respons JSON yang berisi tugas ID, tugas ARN, dan versi perangkat lunak yang diinstal oleh pembaruan:

```
{
    "IotJobId": "GreengrassUpdateJob_c3bd7f36-ee80-4d42-8321-a1da0EXAMPLE",
    "IotJobArn": "arn:aws:iot:region:123456789012:job/GreengrassUpdateJob_c3bd7f36-
ee80-4d42-8321-a1da0EXAMPLE",
    "PlatformSoftwareVersion": "1.9.2"
}
```

Untuk langkah-langkah yang menunjukkan bagaimana menggunakan `create-software-update-job` untuk memperbarui perangkat core, lihat the section called “Buat pembaruan OTA”.

Perintah `create-software-update-job` memiliki parameter berikut:

`--update-targets-architecture`

Arsitektur perangkat core.

Nilai yang benar: `armv7l`, `armv6l`, `x86_64`, atau `aarch64`

`--update-targets`

Core untuk core. Daftar ini dapat berisi ARN dari core individu dan ARN dari grup benda yang anggotanya adalah core. Untuk informasi selengkapnya tentang grup, lihat Grup hal statis di AWS IoT Panduan Developer.

`--update-targets-operating-system`

Sistem operasi perangkat core.

Nilai yang benar: `ubuntu`, `amazon_linux`, `raspbian`, atau `openwrt`

`--software-to-update`

Tentukan apakah perangkat lunak core atau perangkat lunak agen pembaruan OTA harus diperbarui.

Nilai yang valid: `core` or `ota_agent`

`--s3-url-signer-role`

ARN dari IAM role digunakan untuk presign URL Amazon S3 yang tertaut ke artefak pembaruan perangkat lunak AWS IoT Greengrass ini. Kebijakan izin terlampir peran harus mengizinkan `s3:GetObject` tindakan pada bucket dalam target Wilayah AWSs. Peran juga harus mengizinkan `iot.amazonaws.com` untuk menganggap peran sebagai entitas terpercaya. Untuk informasi selengkapnya, lihat the section called “Izin IAM untuk pembaruan OTA”.

`--amzn-client-token`

(Opsional) Token klien yang digunakan untuk membuat permintaan idempoten. Sediakan token unik untuk mencegah duplikat pembaruan yang dibuat karena retries internal.

`--update-agent-log-level`

(Opsional) Tingkat pencatatan untuk pernyataan log yang dihasilkan oleh agen pembaruan OTA. Default-nya adalah `ERROR`.

Nilai yang benar: `NONE`, `TRACE`, `DEBUG`, `VERBOSE`, `INFO`, `WARN`, `ERROR`, atau `FATAL`

> **Note**
>
> `CreateSoftwareUpdateJob` menerima permintaan hanya untuk arsitektur didukung dan kombinasi sistem operasi berikut:

- ubuntu/x86_64
- ubuntu/aarch64
- amazon_linux/x86_64
- raspbian/armv7l
- raspbian/armv6l
- openwrt/aarch64
- openwrt/armv7l

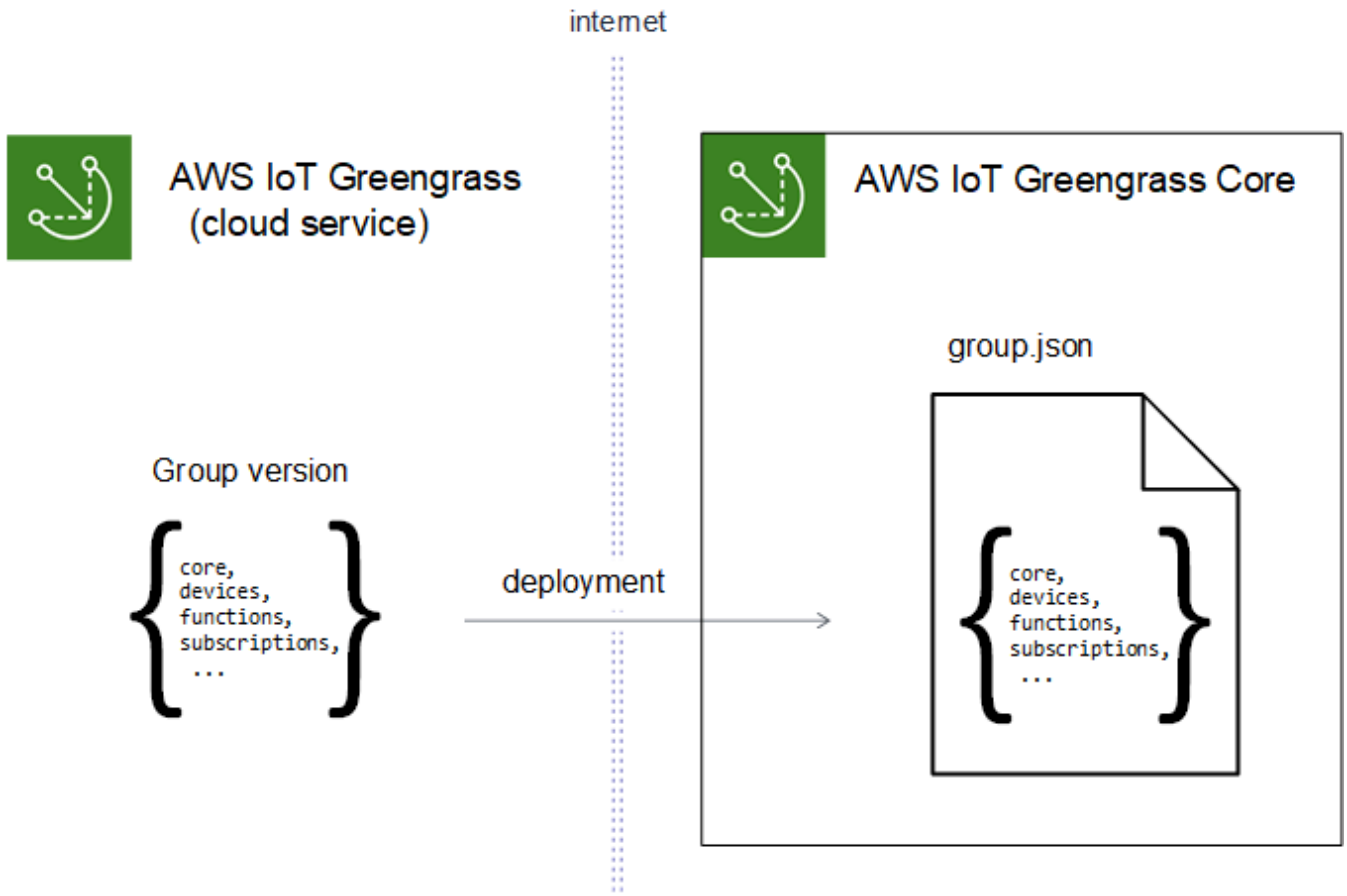
Men-deploy AWS IoT Greengrass grup ke AWS IoT Greengrass core

Gunakan AWS IoT Greengrass grup untuk mengatur entitas di dalam lingkungan edge Anda. Anda juga menggunakan grup untuk mengontrol cara entitas dalam grup berinteraksi satu sama lain dan dengan AWS Cloud. Sebagai contoh, hanya fungsi Lambda di dalam grup yang di-deploy untuk berjalan secara lokal, dan hanya perangkat di dalam grup yang dapat berkomunikasi menggunakan server MQTT lokal.

Sebuah grup harus menyertakan core, yang merupakan AWS IoT perangkat yang menjalankan AWS IoT Greengrass perangkat lunak Core. core bertindak sebagai edge gateway dan menyediakan AWS IoT Core kapabilitas di dalam lingkungan edge. Bergantung pada kebutuhan bisnis Anda, Anda juga dapat menambahkan entitas berikut ke grup:

- Perangkat klien. Diwakili sebagai hal-hal dalam AWS IoT registri. Perangkat ini harus menjalankan FreeRTOS atau menggunakan AWS IoTDevice SDK AWS IoT Greengrassatau Discovery API untuk mendapatkan informasi koneksi untuk inti. Hanya perangkat klien yang merupakan anggota grup yang dapat terhubung ke inti.
- Fungsi Lambda. Aplikasi serverless yang didefinisikan pengguna yang menjalankan kode pada core. Fungsi Lambda ditulis dalam AWS Lambda dan direferensikan dari grup Greengrass. Untuk informasi selengkapnya, lihat Jalankan fungsi Lambda lokal.
- Konektor. Aplikasi serverless standar yang menjalankan kode pada core. Konektor dapat menyediakan integrasi built-in dengan infrastruktur lokal, protokol perangkat, AWS, dan layanan cloud lainnya. Untuk informasi selengkapnya, lihat Integrasikan dengan layanan dan protokol menggunakan konektor.
- Langganan. Mendefinisikan penerbit, pelanggan, dan topik MQTT (atau mata pelajaran) yang berwenang untuk komunikasi MQTT.
- Sumber daya. Referensi ke perangkat dan volume lokal, model machine learning, dan rahasia, digunakan untuk kontrol akses oleh fungsi Lambda Greengrass dan konektor.
- Log. Konfigurasi pencatatan untuk AWS IoT Greengrass komponen sistem dan fungsi Lambda. Untuk informasi selengkapnya, lihat the section called “Pemantauan dengan AWS IoT Greengrass log”.

Anda mengelola grup Greengrass Anda di AWS Cloud lalu men-deploy-nya ke core. Salinan deployment konfigurasi grup untuk file `group.json` pada perangkat core. File ini terletak di *`greengrass-root`*`/ggc/deployments/group`.



> **Note**
>
> Selama deployment, proses Greengrass daemon pada perangkat core berhenti lalu dimulai kembali.

Men-deploy grup dari AWS IoT konsol

Anda dapat men-deploy grup dan mengelola deployment-nya dari halaman konfigurasi grup di AWS IoT konsol.

> **Note**
>
> Untuk membuka halaman ini di konsol, pilih Perangkat Greengrass, lalu Grup (V1), dan kemudian di bawah Grup Greengrass, pilih grup Anda.

Untuk menyebarkan versi grup saat ini

- Dari halaman konfigurasi grup, pilih Deploy.

Untuk melihat riwayat penyebaran grup

Riwayat deployment grup mencakup tanggal dan waktu, versi grup, dan status setiap upaya deployment.

1. Dari halaman konfigurasi grup, pilih tab Deployment.
2. Untuk melihat informasi selengkapnya tentang penerapan, termasuk pesan kesalahan, pilih Penerapan dari AWS IoT konsol, di bawah Perangkat Greengrass.

Untuk menerapkan ulang penyebaran grup

Anda mungkin ingin men-deploy ulang deployment jika deployment saat ini gagal atau kembali ke versi grup yang berbeda.

1. Dari AWS IoT konsol, pilih perangkat Greengrass, lalu pilih Grup (V1).
2. Pilih tab Deployment.
3. Pilih penerapan yang ingin Anda gunakan kembali dan pilih Redeploy.

Untuk mengatur ulang penerapan grup

Anda mungkin ingin mengatur ulang deployment grup untuk memindahkan atau menghapus grup atau untuk menghapus informasi deployment. Untuk informasi selengkapnya, lihat the section called “Atur ulang deployment”.

1. Dari AWS IoT konsol, pilih perangkat Greengrass, lalu pilih Grup (V1).
2. Pilih tab Deployment.
3. Pilih penerapan yang ingin Anda atur ulang dan pilih Atur ulang penerapan.

Men-deploy grup dengan AWS IoT Greengrass API

API AWS IoT Greengrass menyediakan tindakan berikut untuk men-deploy grup AWS IoT Greengrass dan mengelola deployment grup. Anda dapat memanggil tindakan ini dari AWS CLI, AWS IoT Greengrass API, atau AWS SDK.

Tindakan	Deskripsi
CreateDeployment	Membuat sebuah `NewDeployment` atau `Redeployment` deployment. Anda mungkin ingin men-deploy ulang deployment jika deployment saat ini gagal. Atau Anda mungkin ingin men-deploy ulang untuk kembali ke versi grup yang berbeda.
GetDeploymentStatus	Mengembalikan status sebuah deployment: `Building`, `InProgress` , `Success`, atau `Failure`. Anda dapat mengonfigurasi EventBridge acara Amazon untuk menerima pemberitahuan penerapan. Untuk informasi selengkapnya, lihat the section called “Dapatkan notifikasi deployment”.
ListDeployments	Mengembalikan sejarah deployment untuk grup.
ResetDeployments	Mengatur ulang deployment untuk grup. Anda mungkin ingin mengatur ulang deploymen t grup untuk memindahkan atau menghapus grup atau untuk menghapus informasi deployment. Untuk informasi selengkapnya, lihat the section called “Atur ulang deployment”.

> **Note**
>
> Untuk informasi tentang operasi deployment massal, lihat the section called “Buat deployment massal”.

Mendapatkan ID grup

ID grup umumnya digunakan dalam tindakan API. Anda dapat menggunakan ListGroupstindakan untuk menemukan ID grup target dari daftar grup Anda. Sebagai contoh, dalam AWS CLI, gunakan `list-groups` perintah.

```
aws greengrass list-groups
```

Anda juga dapat menyertakan `query` opsi untuk memfilter hasil. Sebagai contoh:

- Untuk mendapatkan grup yang paling baru dibuat:

  ```
  aws greengrass list-groups --query "reverse(sort_by(Groups, &CreationTimestamp))[0]"
  ```

- Untuk mendapatkan grup dengan nama:

  ```
  aws greengrass list-groups --query "Groups[?Name=='MyGroup']"
  ```

  Nama grup tidak perlu unik, sehingga beberapa grup mungkin ditampilkan.

Berikut ini adalah contoh `list-groups` respons. Informasi untuk setiap grup termasuk ID grup (di dalam `Id` properti) dan ID dari versi grup terbaru (dalam `LatestVersion` properti). Untuk mendapatkan ID versi lain untuk grup, gunakan ID grup dengan ListGroupVersions.

> **Note**
>
> Anda juga dapat menemukan nilai-nilai ini di konsol AWS IoT tersebut. ID grup ditampilkan pada halaman Pengaturan grup. ID versi grup ditampilkan di tab Deployment grup.

```
{
```

```
    "Groups": [
        {
            "LatestVersionArn": "arn:aws:us-west-2:123456789012:/greengrass/
groups/00dedaaa-ac16-484d-ad77-c3eedEXAMPLE/versions/4cbc3f07-fc5e-48c4-
a50e-7d356EXAMPLE",
            "Name": "MyFirstGroup",
            "LastUpdatedTimestamp": "2019-11-11T05:47:31.435Z",
            "LatestVersion": "4cbc3f07-fc5e-48c4-a50e-7d356EXAMPLE",
            "CreationTimestamp": "2019-11-11T05:47:31.435Z",
            "Id": "00dedaaa-ac16-484d-ad77-c3eedEXAMPLE",
            "Arn": "arn:aws:us-west-2:123456789012:/greengrass/groups/00dedaaa-
ac16-484d-ad77-c3eedEXAMPLE"
        },
        {
            "LatestVersionArn": "arn:aws:us-west-2:123456789012:/greengrass/
groups/036ceaf9-9319-4716-ba2a-237f9EXAMPLE/versions/8fe9e8ec-64d1-4647-
b0b0-01dc8EXAMPLE",
            "Name": "GreenhouseSensors",
            "LastUpdatedTimestamp": "2020-01-07T19:58:36.774Z",
            "LatestVersion": "8fe9e8ec-64d1-4647-b0b0-01dc8EXAMPLE",
            "CreationTimestamp": "2020-01-07T19:58:36.774Z",
            "Id": "036ceaf9-9319-4716-ba2a-237f9EXAMPLE",
            "Arn": "arn:aws:us-west-2:123456789012:/greengrass/
groups/036ceaf9-9319-4716-ba2a-237f9EXAMPLE"
        },
        ...
    ]
}
```

Jika Anda tidak menentukan Wilayah AWS, AWS CLI memerintahkan untuk menggunakan Wilayah default dari profil Anda. Untuk mengembalikan grup di dalam Wilayah yang berbeda, sertakan opsi *`wilayah`* ini. Sebagai contoh:

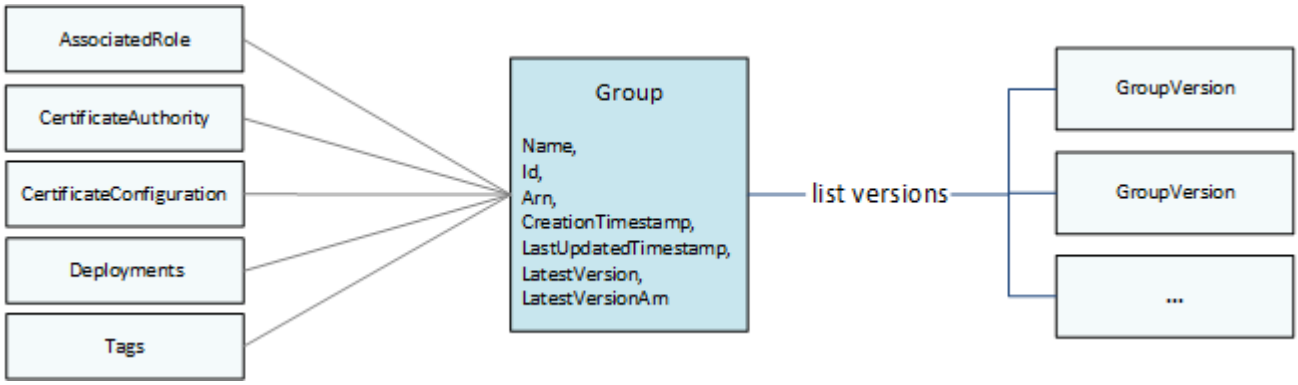
```
aws greengrass list-groups --region us-east-1
```

Ikhtisar tentang AWS IoT Greengrass model objek grup

Ketika memprogram dengan AWS IoT Greengrass API, sangat membantu untuk memahami model objek grup Greengrass.

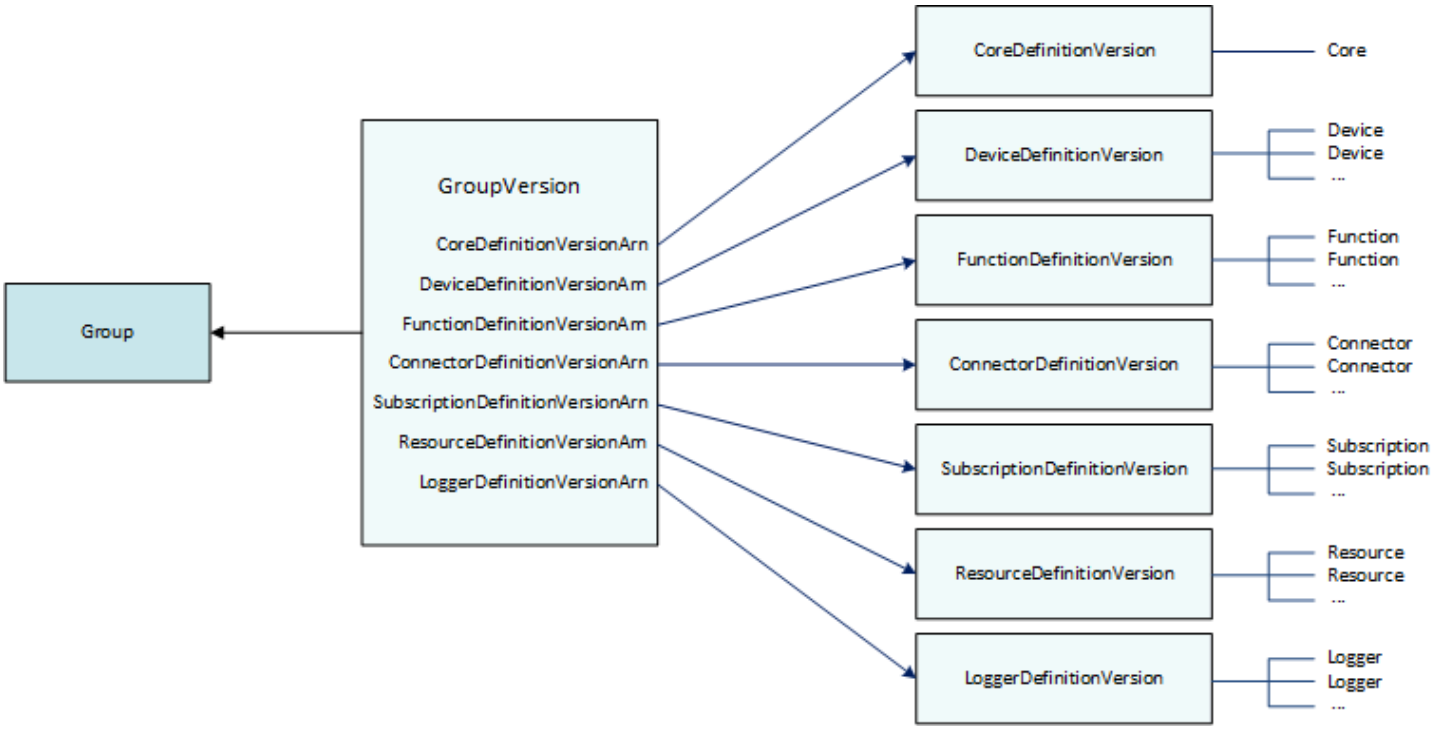
Grup

Di AWS IoT Greengrass API, `Group` objek tingkat atas terdiri dari metadata dan daftar `GroupVersion` objek. `GroupVersion` obyek terkait dengan `Group` oleh ID.



Versi grup

`GroupVersion` objek mendefinisikan keanggotaan grup. Masing-masing `GroupVersion` mereferensi `CoreDefinitionVersion` dan versi komponen lainnya oleh ARN. Referensi ini menentukan entitas mana yang akan disertakan di dalam grup.



Sebagai contoh, untuk menyertakan tiga fungsi Lambda, satu perangkat, dan dua langganan di dalam grup, `GroupVersion` referensi:

- Sebuah `CoreDefinitionVersion` yang berisi core yang dibutuhkan.
- Sebuah `FunctionDefinitionVersion` yang berisi tiga fungsi.
- `DeviceDefinitionVersion`Yang berisi perangkat klien.
- Sebuah `SubscriptionDefinitionVersion` yang berisi dua langganan.

Men-deploy `GroupVersion` ke perangkat core menentukan entitas yang tersedia di lingkungan lokal dan cara mereka dapat berinteraksi.

Komponen grup

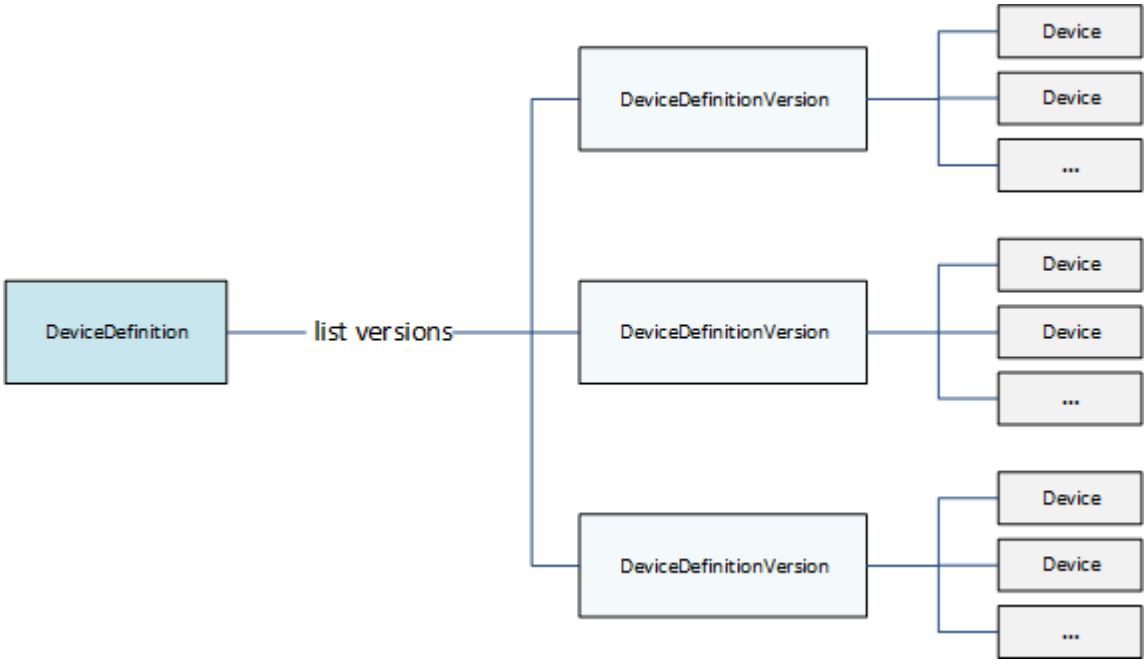
Komponen yang Anda tambahkan ke grup memiliki hirarki tiga tingkat:

- Definisi yang mereferensikan daftar DefinitionVersionobjek dari jenis tertentu. Sebagai contoh, `DeviceDefinition` mereferensikan daftar `DeviceDefinitionVersion` objek.
- A DefinitionVersionyang berisi satu set entitas dari tipe tertentu. Sebagai contoh, `DeviceDefinitionVersion` berisi sebuah daftar dari `Device` objek.
- Entitas individu yang mendefinisikan properti dan perilaku mereka. Misalnya, a `Device` mendefinisikan ARN dari perangkat klien yang sesuai dalam registri, ARN sertifikat AWS IoT perangkatnya, dan apakah bayangan lokalnya disinkronkan secara otomatis dengan cloud.

  Anda dapat menambahkan jenis entitas berikut ke grup:

  - Konektor
  - Inti
  - Perangkat
  - Fungsi
  - Logger
  - Sumber
  - Berlangganan

Contoh berikut `DeviceDefinition` mereferensikan tiga `DeviceDefinitionVersion` objek yang masing-masing berisi beberapa `Device` objek. Hanya satu `DeviceDefinitionVersion` pada suatu waktu digunakan di dalam grup.



Memperbarui grup

Di dalam API AWS IoT Greengrass tersebut, Anda menggunakan versi untuk memperbarui konfigurasi grup. Versi tidak dapat diubah, jadi untuk menambah, menghapus, atau mengubah komponen grup, Anda harus membuat DefinitionVersionobjek yang berisi entitas baru atau yang diperbarui.

Anda dapat mengaitkan DefinitionVersionsobjek baru dengan objek Definisi baru atau yang sudah ada. Sebagai contoh, Anda dapat menggunakan opsi `CreateFunctionDefinition` tindakan untuk membuat `FunctionDefinition` yang mencakup `FunctionDefinitionVersion` sebagai versi awal, atau Anda dapat menggunakan `CreateFunctionDefinitionVersion` tindakan dan mereferensi yang sudah ada `FunctionDefinition`.

Setelah Anda membuat komponen grup, Anda membuat sebuah `GroupVersion` yang berisi semua DefinitionVersionobjek yang ingin Anda sertakan dalam grup. Kemudian, Anda men-deploy `GroupVersion`.

Untuk men-deploy `GroupVersion`, hal itu harus mereferensi `CoreDefinitionVersion` yang berisi persis satu `Core`. Semua entitas yang direferensikan harus menjadi anggota grup. Juga, Peran layanan Greengrass harus dikaitkan dengan Akun AWS Anda dalam Wilayah AWS di mana Anda men-deploy `GroupVersion`.

> **Note**
>
> Tindakan `Update` dalam API digunakan untuk mengubah nama dari `Group` atau komponen Definisi objek.

Memperbarui entitas yang mereferensikan AWS sumber daya

Fungsi Lambda Greengrass dan sumber daya rahasia mendefinisikan properti khusus Greengrass dan juga mereferensi sumber daya AWS yang sesuai. Untuk memperbarui entitas ini, Anda mungkin membuat perubahan pada AWS sumber daya yang sesuai sebagai ganti objek Greengrass Anda. Sebagai contoh, fungsi Lambda mereferensi fungsi di AWS Lambda dan juga mendefinisikan siklus hidup dan properti lainnya yang spesifik untuk grup Greengrass.

- Untuk memperbarui kode fungsi Lambda atau dependensi yang dikemas, buat perubahan Anda di dalam AWS Lambda. Selama deployment grup berikutnya, perubahan ini diambil dari AWS Lambda dan disalin ke lingkungan lokal Anda.
- Untuk memperbarui properti khusus Greengrass, Anda membuat `FunctionDefinitionVersion` yang berisi `Function` properti yang diperbarui.

> **Note**
>
> Fungsi Greengrass Lambda dapat mereferensi fungsi Lambda dengan ARN alias atau ARN versi. Jika Anda mereferensikan ARN alias (disarankan), Anda tidak perlu memperbarui `FunctionDefinitionVersion` (atau `SubscriptionDefinitionVersion`) ketika Anda menerbitkan versi fungsi baru di dalam AWS Lambda. Untuk informasi selengkapnya, lihat the section called “Fungsi referensi dengan alias atau versi”.

Lihat juga

- the section called “Dapatkan notifikasi deployment”
- the section called “Atur ulang deployment”
- the section called “Buat deployment massal”
- Memecahkan Masalah Penerapan

- Referensi API AWS IoT Greengrass Version 1
- AWS IoT Greengrass perintah dalam AWS CLI Referensi Perintah

Dapatkan notifikasi deployment

Amazon EventBridge aturan peristiwa menyediakan Anda pemberitahuan tentang perubahan keadaan untuk deployment grup Greengrass Anda. EventBridge menyampaikan pengaliran acara sistem mendekati waktu nyata yang menjelaskan perubahanAWSsumber daya.AWS IoT Greengrassmengirimkan acara ini ke EventBridge padasetidaknya sekalidasar. Ini berarti bahwa AWS IoT Greengrass mungkin mengirim beberapa salinan dari peristiwa tertentu untuk memastikan pengiriman. Selain itu, pendengar peristiwa Anda mungkin tidak menerima peristiwa dalam urutan peristiwa yang terjadi.

> **Note**
>
> Amazon EventBridge adalah layanan bus peristiwa yang dapat Anda gunakan untuk menghubungkan aplikasi Anda dengan data dari berbagai sumber, seperti Perangkat inti Greengrassdan notifikasi deployment. Untuk informasi selengkapnya, lihat Apa itu Amazon EventBridge?di dalamAmazon EventBridge Panduan Pengguna.

AWS IoT Greengrass mengeluarkan peristiwa ketika deployment grup mengubah keadaan. Anda dapat membuat EventBridge aturan yang berjalan untuk semua transisi keadaan atau transisi ke keadaan yang Anda tentukan. Ketika deployment memasuki keadaan yang memulai aturan, EventBridge memanggil tindakan target yang didefinisikan di dalam aturan. Hal ini mengizinkan Anda untuk mengirim pemberitahuan, menangkap informasi peristiwa, mengambil tindakan korektif, atau menginisiasi peristiwa lain untuk merespon perubahan keadaan. Sebagai contoh, Anda dapat membuat aturan untuk kasus penggunaan berikut:

- Memulai operasi pasca deployment, seperti mengunduh aset dan memberi tahu personil.
- Kirim pemberitahuan setelah deployment berhasil atau gagal.
- Terbitkankan metrik kustom tentang peristiwa deployment.

AWS IoT Greengrass mengeluarkan peristiwa ketika deployment memasuki keadaan berikut: `Building`, `InProgress`, `Success`, dan `Failure`.

> **Note**
>
> Memantau status deployment massal saat ini tidak didukung. Namun, AWS IoT Greengrass mengeluarkan peristiwa perubahan keadaan untuk deployment grup individu yang merupakan bagian dari deployment massal.

Peristiwa perubahan status deployment grup

peristiwa untuk perubahan keadaan deployment menggunakan format berikut:

```
{
    "version":"0",
    "id":" cd4d811e-ab12-322b-8255-EXAMPLEb1bc8",
    "detail-type":"Greengrass Deployment Status Change",
    "source":"aws.greengrass",
    "account":"123456789012",
    "time":"2018-03-22T00:38:11Z",
    "region":"us-west-2",
    "resources":[],
    "detail":{
        "group-id": "284dcd4e-24bc-4c8c-a770-EXAMPLEf03b8",
        "deployment-id": "4f38f1a7-3dd0-42a1-af48-EXAMPLE09681",
        "deployment-type": "NewDeployment|Redeployment|ResetDeployment|
ForceResetDeployment",
        "status": "Building|InProgress|Success|Failure"
    }
}
```

Anda dapat membuat aturan yang berlaku untuk satu atau lebih grup. Anda dapat memfilter aturan berdasarkan satu atau beberapa jenis deployment berikut dan keadaan deployment:

Jenis deployment

- `NewDeployment`. Deployment pertama dari versi grup.
- `ReDeployment`. Sebuah deployment ulang versi grup.
- `ResetDeployment`. Menghapus informasi deployment yang disimpan di AWS Cloud dan pada AWS IoT Greengrass core. Untuk informasi selengkapnya, lihat the section called “Atur ulang deployment”.

- `ForceResetDeployment`. Menghapus informasi deployment yang disimpan di AWS Cloud dan melaporkan keberhasilan tanpa menunggu core untuk merespon. Juga menghapus informasi deployment yang disimpan pada core jika core terhubung atau ketika terhubung selanjutnya.

Keadaan deployment

- `Building`. AWS IoT Greengrass memvalidasi konfigurasi grup dan membangun artefak deployment.
- `InProgress`. Deployment sedang berlangsung pada AWS IoT Greengrass core.
- `Success`. Deployment berhasil.
- `Failure`. Deployment gagal.

Ada kemungkinan bahwa peristiwa dapat diduplikasi atau rusak. Untuk menentukan urutan peristiwa, gunakan properti `time` ini.

> **Note**
>
> AWS IoT Greengrass tidak menggunakan `resources` properti, sehingga selalu kosong.

Prasyarat untuk membuat EventBridge aturan

Sebelum Anda membuat EventBridge aturanAWS IoT Greengrass, lakukan hal berikut:

- Biasakan diri Anda dengan peristiwa, aturan, dan target EventBridge.
- Buat dan konfigurasi target yang dipanggil EventBridge aturan. Aturan dapat memanggil berbagai jenis target, termasuk:
  - Amazon Simple Notification Service (Amazon SNS)
  - AWS Lambda fungsi
  - Amazon Kinesis Video Streams
  - Antrean Amazon Simple Queue Service (Amazon SQS)

Untuk informasi selengkapnya, lihatApa itu Amazon EventBridge?danMemulai dengan Amazon EventBridgedi dalamAmazon EventBridge Panduan Pengguna.

Konfigurasikan pemberitahuan deployment (konsol)

Lakukan langkah-langkah berikut untuk membuat EventBridge aturan yang menerbitkan topik Amazon SNS ketika status deployment berubah untuk grup. Hal ini memungkinkan server web, alamat email, dan pelanggan topik lainnya untuk menanggapi peristiwa tersebut. Untuk informasi selengkapnya, lihatMembuat EventBridge aturan yang memicu peristiwaAWSsumber dayadi dalamAmazon EventBridge Panduan Pengguna.

1. BukaAmazon EventBridgekonsol.
2. Di panel navigasi, pilih Aturan.
3. Pilih Buat aturan.
4. Masukkan nama dan deskripsi untuk aturan.

   Aturan tidak boleh memiliki nama yang sama dengan aturan lain di Wilayah yang sama dan di bus kejadian yang sama.
5. UntukBus peristiwa, pilih bus peristiwa yang ingin Anda kaitkan dengan aturan ini. Jika Anda ingin aturan ini cocok dengan peristiwa yang berasal dari akun Anda, pilihAWSbus kejadian default. Saat layanan AWS di akun Anda menghasilkan peristiwa, layanan tersebut akan selalu masuk ke bus peristiwa default akun Anda.
6. UntukJenis aturan, pilihAturan dengan pola peristiwa.
7. Pilih Selanjutnya.
8. UntukSumber peristiwa, pilihAWSjasa.
9. UntukPola peristiwa, pilihAWSjasa.
10. UntukAWSlayanan, pilih Greengrass.
11. Untuk Tipe peristiwa, pilih Perubahan Status Deployment Greengrass.

    > **Note**
    >
    > ParameterAWSPanggilan API CloudTrailjenis acara didasarkan padaAWS IoT Greengrassintegrasi denganAWS CloudTrail. Anda dapat menggunakan opsi ini untuk membuat aturan yang diinisiasi dengan membaca atau menulis panggilan ke AWS IoT Greengrass API. Untuk informasi selengkapnya, lihat the section called “Mencatat log panggilan API AWS IoT Greengrass dengan AWS CloudTrail”.
12. Pilih keadaan deployment yang menginisiasi pemberitahuan.

- Untuk menerima pemberitahuan untuk semua peristiwa perubahan keadaan, pilih Keadaan apa pun.
- Untuk menerima notifikasi untuk beberapa peristiwa perubahan keadaan saja, pilih Keadaan tertentu, lalu pilih keadaan target.

13. Pilih tipe deployment yang menginisiasi pemberitahuan.

    - Untuk menerima pemberitahuan untuk semua tipe deployment, pilih Keadaan apa pun.
    - Untuk menerima notifikasi untuk beberapa tipe deployment saja, pilih Keadaan tertentu, lalu pilih tipe deployment target.

14. Pilih Selanjutnya.
15. UntukJenis target, pilihAWSlayanan.
16. UntukPilih target, konfigurasikan target Anda. Contoh ini menggunakan topik Amazon SNS, tetapi Anda dapat mengonfigurasi tipe target lain untuk mengirim pemberitahuan.

    a. Untuk Target, pilih topik SNS.

    b. Untuk Topik, pilih topik target Anda.

    c. Pilih Selanjutnya.

17. Di bawahTanda, tentukan tag untuk aturan tersebut atau biarkan kolom tersebut kosong.
18. Pilih Selanjutnya.
19. Tinjau detail aturan dan pilihBuat aturan.

Konfigurasikan pemberitahuan deployment (CLI)

Lakukan langkah-langkah berikut untuk membuat EventBridge aturan yang menerbitkan topik Amazon SNS ketika status deployment berubah untuk grup. Hal ini mengizinkan server web, alamat email, dan pelanggan topik lainnya untuk merespon peristiwa tersebut.

1. Buat aturan .

    - Ganti *id-grup* dengan ID AWS IoT Greengrass grup Anda.

```
aws events put-rule \
  --name TestRule \
```

```
  --event-pattern "{\"source\": [\"aws.greengrass\"], \"detail\": {\"group-id\":
[\"group-id\"]}}"
```

Properti apa pun yang dihilangkan dari pola akan diabaikan.

2. Tambahkan topik sebagai target aturan.

   - Ganti *topik-arn* dengan ARN dari topik Amazon SNS Anda.

```
aws events put-targets \
  --rule TestRule \
  --targets "Id"="1","Arn"="topic-arn"
```

> **Note**
>
> Untuk mengizinkan Amazon EventBridge untuk memanggil topik target Anda, Anda harus menambahkan kebijakan berbasis sumber daya pada topik Anda. Untuk informasi selengkapnya, lihatIzin Amazon SNSdi dalamAmazon EventBridge Panduan Pengguna.

Untuk informasi selengkapnya, lihatPola peristiwa dan pola EventBridgedi dalamAmazon EventBridge Panduan Pengguna.

Konfigurasikan pemberitahuan deployment (AWS CloudFormation)

GunakanAWS CloudFormationtemplat untuk membuat EventBridge aturan yang mengirim pemberitahuan tentang perubahan keadaan untuk deployment grup Greengrass Anda. Untuk informasi selengkapnya, lihatAmazon EventBridge referensi tipe sumber dayadi dalamAWS CloudFormationPanduan Pengguna.

Lihat juga

- Men-deploy AWS IoT Greengrass grup
- Apa itu Amazon EventBridge?di dalamAmazon EventBridge Panduan Pengguna

Atur ulang deployment

Fitur ini tersedia untuk AWS IoT Greengrass core v1.1 lalu.

Anda mungkin ingin mengatur ulang deployment grup ke:

- Hapus grup, seperti saat Anda ingin memindahkan inti grup ke grup lain, atau inti grup telah direimage. Sebelum menghapus grup, Anda harus mengatur ulang penerapan grup untuk menggunakan inti dengan grup Greengrass lain.
- Memindahkan core grup ke grup yang berbeda.
- Mengembalikan grup ke keadaannya sebelum deployment apa pun.
- Menghapus konfigurasi deployment dari perangkat core.
- Menghapus data sensitif dari perangkat core atau dari cloud.
- Men-deploy konfigurasi grup baru ke core tanpa harus mengganti core dengan yang lain di dalam grup saat ini.

> **Note**
>
> Fungsionalitas pengaturan ulang deployment tidak tersedia di AWS IoT Greengrass Perangkat lunak Core v1.0.0. Anda tidak dapat menghapus grup yang telah di-deploy menggunakan v1.0.0.

Operasi pengaturan ulang deployment pertama membersihkan semua informasi deployment yang disimpan di dalam cloud untuk grup tertentu. Kemudian menginstruksikan perangkat core grup untuk membersihkan semua informasi terkait deployment juga (fungsi Lambda, catatan pengguna, database bayangan dan sertifikat server, tetapi bukan sertifikat yang didefinisikan pengguna `config.json` atau sertifikat core Greengrass). Anda tidak dapat memulai pengaturan ulang deployment untuk grup jika grup saat ini memiliki deployment dengan status `In Progress` atau `Building`.

Atur ulang deployment dari AWS IoT konsol

Anda dapat mengatur ulang deployment grup dari halaman konfigurasi grup di AWS IoT konsol.

1. Di panel navigasi AWS IoT konsol, di bawah Kelola, perluas perangkat Greengrass, lalu pilih Grup (V1).
2. Pilih grup target.
3. Dari tab Deployment, pilih Reset deployments.

4. Di kotak dialog Reset deployment for Greengrass Group ini, ketik **`confirm`** untuk setuju, dan pilih Reset deployment.

Atur ulang deployment dengan AWS IoT Greengrass API

Anda dapat menggunakan `ResetDeployments` tindakan di dalam AWS CLI, AWS IoT Greengrass API, atau AWS SDK untuk mengatur ulang deployment. Contoh di dalam topik ini menggunakan CLI.

```
aws greengrass reset-deployments --group-id GroupId [--force]
```

Argumen untuk **`reset-deployments`** perintah CLI:

`--group-id`

ID Grup. Gunakan `list-groups` perintah untuk mendapatkan nilai ini.

`--force`

Opsional. Gunakan parameter ini jika perangkat core grup telah hilang, dicuri, atau dihancurkan. Opsi ini menyebabkan proses pengaturan ulang deployment untuk melaporkan keberhasilan setelah semua informasi deployment di cloud telah dibersihkan, tanpa menunggu perangkat core merespons. Namun, jika perangkat core menjadi aktif, ia juga melakukan operasi pembersihan.

Output dari `reset-deployments` perintah CLI terlihat seperti ini:

```
{
    "DeploymentId": "4db95ef8-9309-4774-95a4-eea580b6ceef",
    "DeploymentArn": "arn:aws:greengrass:us-west-2:106511594199:/greengrass/groups/
b744ed45-a7df-4227-860a-8d4492caa412/deployments/4db95ef8-9309-4774-95a4-eea580b6ceef"
}
```

Anda dapat memeriksa status pengaturan ulang deployment dengan `get-deployment-status` perintah CLI:

```
aws greengrass get-deployment-status --deployment-id DeploymentId --group-id GroupId
```

Argumen untuk **`get-deployment-status`** perintah CLI:

`--deployment-id`

ID deployment.

`--group-id`

ID grup.

Output dari `get-deployment-status` perintah CLI terlihat seperti ini:

```
{
    "DeploymentStatus": "Success",
    "UpdatedAt": "2017-04-04T00:00:00.000Z"
}
```

Mengatur `DeploymentStatus` ke `Building` ketika pengaturan ulang deployment sedang dipersiapkan. Ketika pengaturan ulang deployment siap tetapi AWS IoT Greengrass core belum menangkap pengaturan ulang deployment, `DeploymentStatus` adalah `InProgress`.

Jika operasi pengaturan ulang gagal, informasi kesalahan dikembalikan di dalam respon.

Lihat juga

- Men-deploy AWS IoT Greengrass grup
- ResetDeployments di Referensi AWS IoT Greengrass Version 1 API
- GetDeploymentStatusdi Referensi AWS IoT Greengrass Version 1 API

Buat deployment massal untuk grup

Anda dapat menggunakan panggilan API sederhana untuk deployment sejumlah besar grup Greengrass sekaligus. Deployment ini dipicu dengan tingkat adaptif yang memiliki batas atas tetap.

Tutorial ini menjelaskan cara menggunakan AWS CLI untuk membuat dan memantau deployment grup massal di AWS IoT Greengrass. Contoh deployment massal dalam tutorial ini berisi beberapa grup. Anda dapat menggunakan contoh dalam implementasi Anda untuk menambahkan sebanyak grup yang Anda butuhkan.

Tutorial ini berisi langkah-langkah tingkat tinggi berikut:

1. Buat dan unggah file input deployment massal
2. Membuat dan mengonfigurasi peran eksekusi IAM untuk deployment massal
3. Izinkan akses peran eksekusi ke Bucket S3 Anda
4. Men-deploy grup
5. Uji deployment

Prasyarat

Untuk menyelesaikan tutorial ini, Anda membutuhkan:

- Satu atau lebih grup Greengrass yang dapat di-deploy. Untuk informasi lebih lanjut tentang membuat AWS IoT Greengrass grup and core, lihat Mulai menggunakan AWS IoT Greengrass.
- Menginstal AWS CLI dan mengonfigurasi ke mesin Anda. Untuk informasi lebih lanjut, lihatAWS CLIPanduan Pengguna.
- Bucket S3 yang dibuat di Wilayah AWS sebagai AWS IoT Greengrass. Untuk informasi lebih lanjut, lihatMembuat dan mengonfigurasi bucket S3di dalamPanduan Pengguna Amazon Simple Storage Service.

> **Note**
>
> Saat ini, SSE KMS yang diaktifkan bucket tidak didukung.

Langkah 1: Buat dan unggah file input deployment massal

Dalam langkah ini, Anda membuat file input deployment dan mengunggah ke bucket Amazon S3 Anda. File ini adalah serial, line-delimited file JSON yang berisi informasi tentang setiap grup dalam deployment massal Anda. AWS IoT Greengrass menggunakan informasi ini untuk men-deploy setiap grup atas nama Anda ketika Anda menginisialisasi deployment grup massal.

1. Jalankan perintah berikut untuk mendapatkan `groupId` untuk setiap grup yang ingin Anda gunakan. Anda memasukkan `groupId` ke file input deployment massal Anda sehingga AWS IoT Greengrass dapat mengidentifikasi setiap usaha untuk di-deploy.

> **Note**
>
> Anda juga dapat menemukan nilai-nilai ini di konsol AWS IoT tersebut. ID grup ditampilkan pada halaman Pengaturan grup. ID versi grup ditampilkan pada grupDeploymentTab.

```
aws greengrass list-groups
```

Respons berisi informasi tentang setiap grup di akun AWS IoT Greengrass Anda:

```
{
  "Groups": [
    {
      "Name": "string",
      "Id": "string",
      "Arn": "string",
      "LastUpdatedTimestamp": "string",
      "CreationTimestamp": "string",
      "LatestVersion": "string",
      "LatestVersionArn": "string"
    }
  ],
  "NextToken": "string"
}
```

Jalankan perintah berikut untuk mendapatkan `groupVersionId` dari setiap grup yang ingin Anda men-deploy.

```
list-group-versions --group-id groupId
```

Respons berisi informasi tentang semua versi dalam grup. Buat catatan tentang `Version` nilai untuk versi grup yang ingin Anda gunakan.

```
{
  "Versions": [
    {
      "Arn": "string",
      "Id": "string",
      "Version": "string",
      "CreationTimestamp": "string"
    }
  ],
  "NextToken": "string"
}
```

2. Di terminal komputer Anda atau editor pilihan, buat file,*MyBulkDeploymentInputFile*, dari contoh berikut. File ini berisi informasi tentang masing-masing AWS IoT Greengrass grup untuk dimasukkan dalam deployment massal. Meskipun contoh ini mendefinisikan beberapa grup, untuk tutorial ini, file Anda dapat berisi hanya satu.

   > **Note**
   >
   > Ukuran file ini harus kurang dari 100 MB.

   ```
   {"GroupId":"groupId1", "GroupVersionId":"groupVersionId1",
    "DeploymentType":"NewDeployment"}
   {"GroupId":"groupId2", "GroupVersionId":"groupVersionId2",
    "DeploymentType":"NewDeployment"}
   {"GroupId":"groupId3", "GroupVersionId":"groupVersionId3",
    "DeploymentType":"NewDeployment"}
   ...
   ```

   Setiap catatan (atau baris) berisi objek grup. Setiap objek grup berisi yang sesuai `GroupId` dan `GroupVersionId` dan sebuah `DeploymentType`. Saat ini, AWS IoT Greengrass mendukung `NewDeployment` jenis deployment massal saja.

   Simpan dan tutup file Anda. Buat catatan tentang lokasi file.

3. Gunakan perintah berikut di terminal Anda untuk mengunggah file input Anda ke bucket Amazon S3 Anda. Ganti path file dengan lokasi dan nama file Anda. Untuk informasi, lihat Menambahkan objek ke bucket.

```
aws s3 cp path/MyBulkDeploymentInputFile s3://my-bucket/
```

Langkah 2: Buat dan konfigurasi peran eksekusi IAM

Dalam langkah ini, Anda menggunakan konsol IAM untuk membuat peran eksekusi mandiri. Anda kemudian membangun hubungan kepercayaan antara peran dan AWS IoT Greengrass dan pastikan bahwa pengguna IAM Anda memiliki `PassRole` untuk peran eksekusi Anda. Hal ini mengizinkan AWS IoT Greengrass untuk mengambil peran eksekusi Anda dan membuat deployment atas nama Anda.

1. Gunakan kebijakan berikut untuk membuat peran eksekusi. Dokumen kebijakan ini mengizinkan AWS IoT Greengrass untuk mengakses file input deployment massal Anda ketika membuat setiap deployment atas nama Anda.

   Untuk informasi lebih lanjut tentang membuat IAM role dan mendelegasikan izin, lihat Membuat IAM role.

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Sid": "VisualEditor0",
           "Effect": "Allow",
           "Action": "greengrass:CreateDeployment",
           "Resource": [
               "arn:aws:greengrass:region:accountId:/greengrass/groups/groupId1",
               "arn:aws:greengrass:region:accountId:/greengrass/groups/groupId2",
               "arn:aws:greengrass:region:accountId:/greengrass/groups/groupId3",
               ...
           ]
       }
   ]
}
```

> **Note**
>
> Kebijakan ini harus memiliki sumber daya untuk setiap grup atau grup versi dalam file input deployment massal untuk di-deploy oleh AWS IoT Greengrass. Untuk mengizinkan akses ke semua grup, untuk `Resource`, tentukan tanda bintang:
>
> ```
> "Resource": ["*"]
> ```

2. Modifikasi hubungan kepercayaan untuk peran eksekusi Anda untuk menyertakan AWS IoT Greengrass. Ini mengizinkan AWS IoT Greengrass untuk menggunakan peran eksekusi Anda dan izin yang melekat padanya. Untuk informasi, lihat Mengedit hubungan kepercayaan untuk peran yang ada.

   Kami merekomendasikan Anda juga memasukkan`aws:SourceArn`dan`aws:SourceAccount`kunci konteks kondisi global dalam kebijakan kepercayaan Anda untuk membantu mencegah*deputi bingung*masalah keamanan. Kunci konteks kondisi membatasi akses untuk mengizinkan hanya permintaan yang berasal dari akun tertentu dan ruang kerja Greengrass. Untuk informasi lebih lanjut tentang masalah deputi yang membingungkan, lihatCross-service bingung wakil pencegahan.

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "",
         "Effect": "Allow",
         "Principal": {
           "Service": "greengrass.amazonaws.com"
         },
         "Action": "sts:AssumeRole",
         "Condition": {
           "StringEquals": {
             "aws:SourceAccount": "account-id"
           },
           "ArnLike": {
             "aws:SourceArn": "arn:aws:greengrass:region:account-id:*"
           }
         }
   ```

```
    }
  ]
}
```

3. Berikan IAM `PassRole` izin untuk peran eksekusi ke pengguna IAM Anda. Pengguna IAM ini adalah salah satu yang digunakan untuk memulai deployment massal. `PassRole` mengizinkan pengguna IAM Anda untuk meneruskan peran eksekusi Anda ke AWS IoT Greengrass untuk digunakan. Untuk informasi lebih lanjut, lihat Memberikan izin pengguna untuk meneruskan peran ke AWS layanan.

   Gunakan contoh berikut untuk memperbarui kebijakan IAM yang dilampirkan pada peran eksekusi yang dilampirkan pada peran eksekusi Anda. Memodifikasi contoh ini, jika perlu.

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Sid": "Stmt1508193814000",
           "Effect": "Allow",
           "Action": [
               "iam:PassRole"
           ],
           "Resource": [
               "arn:aws:iam::account-id:user/executionRoleArn"
           ]
           "Condition": {
               "StringEquals": {
                   "iam:PassedToService": "greengrass.amazonaws.com"
               }
           }
       }
   ]
}
```

Langkah 3: Izinkan akses peran eksekusi ke Bucket S3 Anda

Untuk memulai deployment massal Anda, peran eksekusi Anda harus dapat membaca file input deployment massal Anda dari bucket Amazon S3 Anda. Lampirkan kebijakan contoh berikut untuk bucket Amazon S3 Anda sehingga `GetObject` izin dapat diakses ke peran eksekusi Anda.

Untuk informasi lebih lanjut, lihat Cara menambahkan kebijakan bucket S3?

```
{
    "Version": "2008-10-17",
    "Id": "examplePolicy",
    "Statement": [
        {
            "Sid": "Stmt1535408982966",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "executionRoleArn"
                ]
            },
            "Action": "s3:GetObject",
            "Resource":
            "arn:aws:s3:::my-bucket/objectKey"
        }
    ]
}
```

Anda dapat menggunakan perintah berikut di terminal Anda untuk memeriksa kebijakan bucket:

```
aws s3api get-bucket-policy --bucket my-bucket
```

Note

Anda dapat langsung memmodifikasi peran eksekusi Anda untuk memberikan izin ke bucket Amazon S3 `GetObject` sebagai gantinya. Untuk melakukannya, lampirkan kebijakan berikut untuk peran eksekusi Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-bucket/objectKey"
```

```
        }
    ]
}
```

Langkah 4: Men-deploy grup

Dalam langkah ini, Anda memulai operasi deployment massal untuk semua versi grup yang dikonfigurasi dalam file input deployment massal Anda. Tindakan deployment untuk setiap versi grup Anda adalah jenis `NewDeploymentType`.

> **Note**
>
> Anda tidak dapat memanggil StartBulkDeployment sementara deployment massal lain dari akun yang sama masih berjalan. Permintaan ditolak.

1. Gunakan perintah berikut untuk memulai deployment massal.

   Kami merekomendasikan Anda menyertakan `X-Amzn-Client-Token` token di setiap StartBulkDeployment permintaan. Permintaan ini idempoten sehubungan dengan token dan parameter permintaan. Token ini dapat berupa string yang unik dan sensitif hingga 64 karakter ASCII.

   ```
   aws greengrass start-bulk-deployment --cli-input-json "{
            "InputFileUri":"URI of file in S3 bucket",
            "ExecutionRoleArn":"ARN of execution role",
            "AmznClientToken":"your Amazon client token"
            }"
   ```

   Perintah harus menghasilkan kode status sukses `200`, bersama dengan respon berikut:

   ```
   {
     "bulkDeploymentId": UUID
   }
   ```

Membuat catatan ID deployment massal. Hal ini dapat digunakan untuk memeriksa status deployment massal Anda.

> **Note**
>
> Meskipun operasi deployment massal saat ini tidak didukung, Anda dapat membuat Amazon EventBridge aturan acara untuk mendapatkan notifikasi tentang perubahan status deployment untuk masing-masing grup. Untuk informasi selengkapnya, lihat the section called “Dapatkan notifikasi deployment”.

2. Gunakan perintah berikut untuk memeriksa status deployment massal Anda.

```
aws greengrass get-bulk-deployment-status --bulk-deployment-id 1234567
```

Perintah tersebut seharusnya mengembalikan kode status sukses dari `200` selain muatan JSON informasi:

```
 {
  "BulkDeploymentStatus": Running,
  "Statistics": {
     "RecordsProcessed": integer,
     "InvalidInputRecords": integer,
     "RetryAttempts": integer
  },
  "CreatedAt": "string",
  "ErrorMessage": "string",
  "ErrorDetails": [
    {
      "DetailedErrorCode": "string",
      "DetailedErrorMessage": "string"
    }
  ]
}
```

`BulkDeploymentStatus` Berisi status eksekusi massal saat ini. Eksekusi dapat memiliki satu dari enam status yang berbeda:

- `Initializing`. Permintaan deployment massal telah diterima, dan eksekusi sedang mempersiapkan untuk mulai.
- `Running`. Eksekusi deployment massal telah dimulai.
- `Completed`. Eksekusi deployment massal telah selesai memproses semua catatan.
- `Stopping`. Eksekusi deployment massal telah menerima perintah untuk berhenti dan akan mengakhiri segera. Anda tidak dapat memulai deployment massal baru sementara deployment sebelumnya di `Stopping` keadaan.
- `Stopped`. Eksekusi deployment massal telah secara manual berhenti.
- `Failed`. Eksekusi deployment massal mengalami kesalahan dan dihentikan. Anda dapat menemukan rincian kesalahan dalam `ErrorDetails` bidang.

Muatan JSON juga mencakup informasi statistik tentang kemajuan deployment massal. Anda dapat menggunakan informasi ini untuk menentukan berapa banyak grup telah diproses dan berapa banyak telah gagal. Informasi statistik meliputi:

- `RecordsProcessed`: Jumlah grup catatan yang dicoba.
- `InvalidInputRecords`: Jumlah total catatan yang mengembalikan kesalahan yang tidak dapat dicoba lagi. Sebagai contoh, hal ini dapat terjadi jika catatan grup dari file input menggunakan format yang tidak valid atau menentukan versi grup yang tidak ada, atau jika eksekusi tidak memberikan izin untuk deployment versi grup atau grup.
- `RetryAttempts`: Jumlah upaya deployment yang mengembalikan kesalahan yang dapat dicoba lagi. Sebagai contoh, coba lagi dipicu jika upaya untuk men-deploy grup mengembalikan kesalahan perlambatan. Deployment grup dapat dicoba ulang hingga lima kali.

Dalam kasus kegagalan eksekusi deployment massal, muatan ini juga mencakup `ErrorDetails` bagian yang dapat digunakan untuk pemecahan masalah. Ini berisi informasi tentang penyebab kegagalan eksekusi.

Anda dapat secara berkala memeriksa status deployment massal untuk mengkonfirmasi bahwa kemajuan seperti yang diharapkan. Setelah deployment selesai, `RecordsProcessed` harus sama dengan jumlah grup deployment dalam file input deployment massal Anda. Hal ini menunjukkan bahwa setiap catatan telah diproses.

Langkah 5: Uji deployment

Gunakan ListBulkDeployments perintah untuk menemukan ID deployment massal Anda.

```
aws greengrass list-bulk-deployments
```

Perintah ini akan menampilkan daftar semua deployment massal dari sebagian besar hingga paling tidak terbaru, termasuk `BulkDeploymentId`.

```
{
  "BulkDeployments": [
    {
      "BulkDeploymentId": 1234567,
      "BulkDeploymentArn": "string",
      "CreatedAt": "string"
    }
  ],
  "NextToken": "string"
}
```

Sekarang panggilan ListBulkDeploymentDetailedReports perintah untuk mengumpulkan informasi rinci tentang setiap deployment.

```
aws greengrass list-bulk-deployment-detailed-reports --bulk-deployment-id 1234567
```

Perintah tersebut seharusnya mengembalikan kode status sukses dari `200` bersama dengan muatan JSON informasi:

```
{
  "BulkDeploymentResults": [
    {
      "DeploymentId": "string",
      "GroupVersionedArn": "string",
```

```
      "CreatedAt": "string",
      "DeploymentStatus": "string",
      "ErrorMessage": "string",
      "ErrorDetails": [
        {
          "DetailedErrorCode": "string",
          "DetailedErrorMessage": "string"
        }
      ]
    }
  ],
  "NextToken": "string"
}
```

Muatan ini biasanya berisi daftar paginasi dari setiap deployment dan status deployment dari yang terbaru hingga yang terbaru. Hal ini juga berisi informasi lebih lanjut dalam hal kegagalan eksekusi deployment massal. Sekali lagi, jumlah total deployment terdaftar harus sama dengan jumlah grup yang Anda identifikasi dalam file input deployment massal Anda.

Informasi yang dikembalikan dapat berubah sampai deployment berada dalam keadaan terminal (keberhasilan atau kegagalan). Anda dapat memanggil perintah ini secara berkala sampai saat itu.

Pemecahan masalah deployment massal

Jika deployment massal tidak berhasil, Anda dapat mencoba langkah-langkah pemecahan masalah berikut ini. Jalankan perintah di terminal Anda.

Memecahkan masalah kesalahan file input

Deployment massal dapat gagal dalam hal kesalahan sintaks dalam file input deployment massal. Ini mengembalikan status deployment massal `Failed` dengan pesan kesalahan yang menunjukkan nomor baris kesalahan validasi pertama. Ada empat kemungkinan kesalahan:

- ```
  InvalidInputFile: Missing GroupId at line number: line number
  ```

  Kesalahan ini mengindikasikan bahwa baris file input yang diberikan tidak dapat mendaftarkan parameter yang ditentukan. Parameter yang hilang mungkin adalah `GroupId` dan `GroupVersionId`.
  ```

- ```
  InvalidInputFile: Invalid deployment type at line number : line number. Only valid
   type is 'NewDeployment'.
  ```

  Kesalahan ini mengindikasikan bahwa baris file input yang diberikan mencantumkan jenis deployment yang tidak valid. Pada saat ini, satu-satunya jenis deployment yang didukung adalah `NewDeployment`.

- ```
  Line %s is too long in S3 File. Valid line is less than 256 chars.
  ```

  Kesalahan ini menunjukkan bahwa baris file input yang diberikan terlalu panjang dan harus dipersingkat.

- ```
  Failed to parse input file at line number: line number
  ```

  Kesalahan ini menunjukkan bahwa baris file input yang diberikan tidak dianggap json yang valid.

Periksa deployment massal bersamaan

Anda tidak dapat memulai deployment massal baru sementara yang lain masih berjalan atau dalam keadaan non-terminal. Ini dapat mengakibatkan `Concurrent Deployment Error`. Anda dapat menggunakan ListBulkDeployments perintah untuk memverifikasi bahwa deployment massal tidak sedang berjalan. Perintah ini mencantumkan deployment massal Anda dari yang terbaru hingga yang terbaru.

```
{
  "BulkDeployments": [
    {
      "BulkDeploymentId": BulkDeploymentId,
      "BulkDeploymentArn": "string",
      "CreatedAt": "string"
    }
  ],
  "NextToken": "string"
}
```
```

Gunakan `BulkDeploymentId` deployment massal terdaftar pertama untuk menjalankan GetBulkDeploymentStatus perintah. Jika penggunaan massal terbaru Anda dalam keadaan berjalan (`Initializing` atau `Running`), gunakan perintah berikut untuk menghentikan deployment massal.

```
aws greengrass stop-bulk-deployment --bulk-deployment-id BulkDeploymentId
```

Tindakan ini menghasilkan status `Stopping` sampai deployment adalah `Stopped`. Setelah deployment telah mencapai `Stopped` status, Anda dapat memulai deployment massal baru.

MEMERIKSA ErrorDetails

Jalankan `GetBulkDeploymentStatus` perintah untuk mengembalikan muatan JSON yang berisi informasi tentang kegagalan eksekusi deployment massal.

```
  "Message": "string",
  "ErrorDetails": [
    {
      "DetailedErrorCode": "string",
      "DetailedErrorMessage": "string"
    }
  ]
```

Ketika keluar dengan kesalahan, `ErrorDetails` muatan JSON yang dikembalikan oleh panggilan ini berisi informasi lebih lanjut tentang kegagalan eksekusi deployment massal. Kode status kesalahan dalam `400` seri, sebagai contoh, menunjukkan kesalahan input, baik dalam parameter input atau dependensi pemanggil.

Periksa AWS IoT Greengrass log core

Anda dapat memecahkan masalah dengan melihat AWS IoT Greengrass catatan core. Gunakan perintah berikut untuk melihat `runtime.log`:

```
cd /greengrass/ggc/var/log
sudo cat system/runtime.log | more
```

Untuk informasi lebih lanjut tentang AWS IoT Greengrass catatan, lihat Pemantauan dengan AWS IoT Greengrass log.

Lihat juga

Untuk informasi lebih lanjut, lihat sumber daya berikut:

- Men-deploy AWS IoT Greengrass grup
- Perintah API Amazon S3 dalam AWS CLI Referensi Perintah
- AWS IoT Greengrass perintah dalam AWS CLI Referensi Perintah

Jalankan fungsi Lambda pada AWS IoT Greengrass core

AWS IoT Greengrass menyediakan lingkungan waktu aktif Lambda terkontainerisasi untuk kode yang ditetapkan pengguna yang Anda tulis di AWS Lambda. Fungsi Lambda yang di-deploy ke AWS IoT Greengrass core berjalan di waktu aktif Lambda lokal Core. Fungsi Lambda lokal dapat dipicu oleh peristiwa lokal, pesan dari cloud, dan sumber lain, yang membawa fungsionalitas komputasi lokal ke perangkat klien. Sebagai contoh, Anda dapat menggunakan fungsi Greengrass Lambda untuk memfilter data perangkat sebelum mengirimkan data ke cloud.

Untuk men-deploy fungsi Lambda ke core, Anda menambahkan fungsi ke grup Greengrass (dengan referensi fungsi Lambda yang ada), mengonfigurasi pengaturan khusus grup untuk fungsi, dan kemudian men-deploy grup. Jika fungsi mengakses AWS layanan, Anda juga harus menambahkan izin yang diperlukan untuk peran grup Greengrass.

Anda dapat mengonfigurasi parameter yang menentukan bagaimana fungsi Lambda berjalan, termasuk izin, isolasi, batas memori, dan banyak lagi. Untuk informasi selengkapnya, lihat the section called “Mengontrol eksekusi fungsi Greengrass Lambda”.

> **Note**
>
> Pengaturan ini juga memungkinkan untuk menjalankan AWS IoT Greengrass dalam kontainer Docker. Untuk informasi selengkapnya, lihat the section called “Jalankan AWS IoT Greengrass di kontainer Docker”.

Tabel berikut mencantumkan data yang didukung AWS Lambda waktu aktif dan versi AWS IoT Greengrass perangkat lunak Core yang dapat mereka jalankan.

Bahasa atau platform	Versi GGC
Python 3.8	1.11
Python 3.7	1.9 atau yang lebih baru
Python 2.7 *	1.0 atau yang lebih baru
Java 8	1.1 atau yang lebih baru

Bahasa atau platform	Versi GGC
Node.js 12.x *	1.10 atau yang lebih baru
Node.js 8.10 *	1.9 atau yang lebih baru
Node.js 6.10 *	1.1 atau yang lebih baru
C, C ++	1.6 atau yang lebih baru

* Anda dapat menjalankan fungsi Lambda yang menggunakan waktu aktif ini pada versi yang didukung dari AWS IoT Greengrass, tetapi Anda tidak dapat membuatnya di AWS Lambda. Jika runtime pada perangkat Anda berbeda dari runtime AWS Lambda yang ditentukan untuk fungsi tersebut, Anda dapat memilih runtime sendiri dengan menggunakan in. `FunctionRuntimeOverride` `FunctionDefintionVersion` Untuk informasi lebih lanjut, lihat CreateFunctionDefinition. Untuk informasi selengkapnya tentang runtime yang didukung, lihat Kebijakan dukungan Runtime di Panduan AWS LambdaPengembang.

SDK untuk fungsi Greengrass Lambda

AWS menyediakan tiga SDK yang dapat digunakan oleh fungsi Greengrass Lambda yang berjalan pada AWS IoT Greengrass core. SDK ini terkandung dalam paket yang berbeda, sehingga fungsi dapat menggunakannya secara bersamaan. Untuk menggunakan SDK dalam fungsi Greengrass Lambda, memasukkannya ke dalam paket deployment fungsi Lambda yang Anda upload ke AWS Lambda.

AWS IoT GreengrassSDK inti

Mengaktifkan fungsi Lambda lokal untuk berinteraksi dengan core untuk:

- Pertukaran pesan MQTT dengan AWS IoT Core.
- Tukar pesan MQTT dengan konektor, perangkat klien, dan fungsi Lambda lainnya di grup Greengrass.
- Berinteraksi dengan layanan bayangan lokal.
- Meminta fungsi Lambda lokal lainnya.
- Akses Sumber daya rahasia.
- Berinteraksi dengan Pengelola aliran.

AWS IoT Greengrassmenyediakan AWS IoT Greengrass Core SDK dalam bahasa dan platform berikut di GitHub.

- AWS IoT GreengrassCore SDK for Java
- AWS IoT GreengrassCore SDK untuk Node.js
- AWS IoT GreengrassCore SDK untuk Python
- AWS IoT GreengrassCore SDK untuk C

Untuk menyertakan AWS IoT Greengrass Core SDK depedensi dalam paket deployment fungsi Lambda:

1. Mengunduh bahasa atau platform AWS IoT Greengrass paket Core SDK yang cocok dengan waktu aktif fungsi Lambda Anda.
2. Unzip paket unduhan untuk mendapatkan SDK. SDK adalah `greengrasssdk` folder.
3. Sertakan `greengrasssdk` dalam paket deployment fungsi Lambda yang berisi kode fungsi Anda. Ini adalah paket yang Anda unggah ke AWS Lambda ketika Anda membuat fungsi Lambda.

StreamManagerClient

Hanya AWS IoT Greengrass Core SDK core berikut yang dapat digunakan untuk operasi pengelola pengaliran ini:

- SDK Java (v1.4.0 atau yang lebih baru)
- Python SDK (v1.5.0 atau yang lebih baru)
- Node.js SDK (v1.6.0 atau yang lebih baru)

Untuk menggunakan AWS IoT Greengrass Core SDK for Python untuk berinteraksi dengan pengelola pengaliran, Anda harus menginstal Python 3.7 atau yang lebih baru. Anda juga harus mengmasang dependensi untuk disertakan dalam paket deployment fungsi Python Lambda Anda:

1. Arahkan ke direktori SDK yang berisi file `requirements.txt` ini. File ini berisi daftar dependensi.
2. Instal dependensi SDK. Misalnya, jalankan perintah berikut `pip` untuk memasangnya di direktori ketika ini:

```
pip install --target . -r requirements.txt
```

Instal AWS IoT Greengrass Core SDK for Python pada perangkat core

Jika Anda menjalankan fungsi Python Lambda, Anda juga dapat menggunakan `pip` untuk menginstal AWS IoT Greengrass Core SDK for Python pada perangkat core. Kemudian Anda dapat men-deploy fungsi Anda tanpa memasukkan SDK dalam paket deployment fungsi Lambda. Untuk informasi lebih lanjut, lihat greengrasssdk.

Dukungan ini ditujukan untuk core dengan pengukuran terbatas. Kami merekomendasikan Anda menyertakan SDK dalam paket deployment fungsi Lambda Anda bila memungkinkan.

AWS IoT GreengrassSDK Machine Learning

Mengaktifkan fungsi Lambda lokal untuk mengonsumsi model machine learning (ML) yang di-deploy ke Greengrass core sebagai sumber daya ML. Fungsi Lambda dapat menggunakan SDK untuk memanggil dan berinteraksi dengan layanan kesimpulan lokal yang di-deploy ke core sebagai konektor. fungsi Lambda dan konektor ML juga dapat menggunakan SDK untuk mengirim data ke konektor ML Feedback untuk mengunggah dan menerbitkan. Untuk informasi lebih lanjut, termasuk contoh kode yang menggunakan SDK, lihat the section called “Klasifikasi Citra ML”, the section called “Deteksi Objek ML”, dan the section called “Umpan balik ML”.

Tabel berikut mencantumkan bahasa yang didukung atau platform untuk versi SDK dan versi perangkat lunak AWS IoT Greengrass core yang dapat mereka jalankan.

Versi SDK	Bahasa atau platform	Versi GGC yang diperlukan	Changelog
1.1.0	Python 3.7 atau 2.7	1.9.3 atau yang lebih baru	Ditambahkan dukungan Python 3.7 dan klien `feedback` baru.

Versi SDK	Bahasa atau platform	Versi GGC yang diperlukan	Changelog
1.0.0	Python 2.7	1.7 atau yang lebih baru	Pelepasan awal.

Untuk informasi unduhan, lihat the section called “AWS IoT Greengrass Perangkat lunak ML SDK”.

AWS SDK

Mengaktifkan fungsi Lambda lokal untuk membuat panggilan langsung ke layanan AWS ini, seperti Amazon S3, DynamoDB, AWS IoT, dan AWS IoT Greengrass. Untuk menggunakan sebuah perangkat AWS SDK dalam fungsi Greengrass Lambda, Anda harus memasukkannya dalam paket deployment Anda. Saat Anda menggunakan AWS SDK dalam paket yang sama dengan AWS IoT Greengrass Core SDK, pastikan bahwa fungsi Lambda Anda menggunakan namespace yang benar. Fungsi Greengrass Lambda tidak dapat berkomunikasi dengan layanan cloud ketika core-nya offline.

Mengunduh SDK AWS dari Pusat Sumber Daya Memulai.

Untuk informasi lebih lanjut tentang cara membuat paket deployment, lihat the section called “Buat dan paketkan fungsi Lambda” dalam tutorial Memulai atau Membuat paket deployment dalam AWS Lambda Panduan Developer.

Memigrasi fungsi Lambda berbasis cloud

SDK Core AWS IoT Greengrass mengikuti model pemrograman AWS SDK, yang membuatnya mudah untuk port fungsi Lambda yang dikembangkan untuk cloud pada fungsi Lambda yang berjalan dengan AWS IoT Greengrass core.

Sebagai contoh, fungsi Python Lambda berikut menggunakan AWS SDK for Python (Boto3) untuk mempublikasikan pesan ke topik `some/topic` di cloud:

```
import boto3

iot_client = boto3.client("iot-data")
```

```
response = iot_client.publish(
    topic="some/topic", qos=0, payload="Some payload".encode()
)
```

Untuk port fungsi pada AWS IoT Greengrass core, di pernyataan `import` dan `client` inisialisasi, mengubah nama modul `boto3` ke `greengrasssdk`, seperti yang ditunjukkan dalam contoh berikut:

```
import greengrasssdk

iot_client = greengrasssdk.client("iot-data")
iot_client.publish(topic="some/topic", qos=0, payload="Some payload".encode())
```

> **Note**
>
> SDK Core AWS IoT Greengrass mendukung pengiriman pesan MQTT dengan QoS = 0 saja. Untuk informasi selengkapnya, lihat the section called “Kualitas layanan pesan”.

Kesamaan antara model pemrograman juga memungkinkan Anda untuk mengembangkan fungsi Lambda Anda di cloud dan kemudian memigrasikannya ke AWS IoT Greengrass dengan sedikit usaha. Lambda yang dapat di eksekusi tidak berjalan di cloud, sehingga Anda tidak dapat menggunakan AWS SDK untuk mengembangkannya di cloud sebelum deployment.

Fungsi Referensi Lambda dengan alias atau versi

Grup Greengrass dapat referensi fungsi Lambda dengan alias (direkomendasikan) atau dengan versi. Menggunakan alias membuatnya lebih mudah untuk mengelola pembaruan kode karena Anda tidak perlu mengubah tabel langganan atau definisi grup ketika kode fungsi diperbarui. Sebaliknya, Anda hanya mengarahkan alias ke versi fungsi baru. Alias menyelesaikan ke nomor versi selama deployment grup. Ketika Anda menggunakan alias, versi yang diselesaikan telah diperbarui ke versi di mana alias menunjuk ke ketika deployment.

AWS IoT Greengrass tidak mendukung alias Lambda untuk versi $TERBARU ini. $TERBARU versi tidak terikat untuk tetap, versi fungsi diterbitkan dan dapat diubah setiap ketika, yang bertentangan dengan prinsip AWS IoT Greengrass versi tetap.

Sebuah praktek umum untuk menjaga fungsi Greengrass Lambda Anda diperbarui dengan perubahan kode adalah dengan menggunakan alias bernama **`PRODUCTION`** dalam grup Greengrass dan langganan Anda. Ketika Anda mempromosikan versi baru dari fungsi Lambda Anda ke dalam

produksi, arahkan alias ke versi stabil terbaru dan kemudian redeploy grupnya. Anda juga dapat menggunakan metode ini untuk kembali ke versi sebelumnya.

Mengontrol eksekusi fungsi Greengrass Lambda dengan menggunakan konfigurasi grup khusus

AWS IoT Greengrass menyediakan manajemen berbasis cloud dari fungsi Greengrass Lambda. Meskipun kode fungsi Lambda dan dependensi dikelola menggunakan AWS Lambda, Anda dapat mengonfigurasi bagaimana fungsi Lambda berperilaku ketika berjalan dalam grup Greengrass.

Pengaturan konfigurasi grup khusus

AWS IoT Greengrass menyediakan pengaturan konfigurasi grup khusus berikut untuk fungsi Greengrass Lambda.

Pengguna sistem dan grup

Identitas akses yang digunakan untuk menjalankan fungsi Lambda. Secara default, fungsi Lambda dijalankan sebagai identitas akses default. Biasanya, ini adalah standar akun sistem AWS IoT Greengrass (ggc_user dan ggc_group). Anda dapat mengubah pengaturan dan memilih ID pengguna dan ID grup yang memiliki izin yang diperlukan untuk menjalankan fungsi Lambda. Anda dapat mengganti kedua UID dan GID atau hanya satu jika Anda membiarkan bidang lain kosong. Pengaturan ini memberi Anda kontrol yang lebih terperinci atas akses ke sumber daya perangkat. Kami menyarankan Anda mengonfigurasi perangkat keras Greengrass Anda dengan batas sumber daya yang sesuai, izin file, dan kuota disk untuk pengguna dan grup yang izinnya digunakan untuk menjalankan fungsi Lambda.

Fitur ini tersedia untuk AWS IoT Greengrass Core v1.7 dan yang lebih baru.

> **Important**
>
> Kami merekomendasikan Anda menghindari menjalankan fungsi Lambda sebagai root kecuali benar-benar diperlukan. Menjalankan sebagai root meningkatkan risiko berikut:
>
> - Risiko perubahan yang tidak diinginkan, seperti secara tidak sengaja menghapus file kritis.
> - Risiko terhadap data dan perangkat Anda dari individu jahat.
> - Risiko kontainer lolos ketika kontainer Docker berjalan dengan `--net=host` dan `UID=EUID=0`.

Jika Anda perlu menjalankan sebagai root, Anda harus memperbarui konfigurasi AWS IoT Greengrass untuk mengaktifkannya. Untuk informasi selengkapnya, lihat the section called “Menjalankan fungsi Lambda sebagai root”.

ID pengguna sistem (nomor)

ID pengguna untuk pengguna berizin yang diperlukan untuk menjalankan fungsi Lambda. Pengaturan ini hanya tersedia jika Anda memilih untuk menjalankan sebagai ID pengguna/ID grup lain. Anda dapat menggunakan perintah getent passwd pada AWS IoT Greengrass untuk mencari ID pengguna yang ingin Anda gunakan untuk menjalankan fungsi Lambda.

Jika Anda menggunakan UID yang sama untuk menjalankan proses dan fungsi Lambda pada perangkat core Greengrass, peran grup Greengrass Anda dapat memberikan proses kredensial sementara. Proses dapat menggunakan kredensial sementara di Greengrass core deployment.

ID grup sistem (nomor)

ID grup untuk grup berizin yang diperlukan untuk menjalankan fungsi Lambda. Pengaturan ini hanya tersedia jika Anda memilih untuk menjalankan sebagai ID pengguna/ID grup lain. Anda dapat menggunakan perintah getent group pada AWS IoT Greengrass untuk mencari ID grup yang ingin Anda gunakan untuk menjalankan fungsi Lambda.

Kontainerisasi fungsi Lambda

Pilih apakah fungsi Lambda berjalan dengan kontainerisasi default untuk grup, atau menentukan kontainerisasi yang harus selalu digunakan untuk fungsi Lambda ini.

Mode kontainerisasi dari sebuah fungsi Lambda menentukan tingkat isolasi.

- Fungsi Lambda dalam kontainer dijalankan dalam mode kontainer Greengrass ini. Fungsi Lambda berjalan di lingkungan waktu aktif terisolasi (atau namespace) di dalam kontainer AWS IoT Greengrass ini.
- Fungsi Lambda tanpa kontainer berjalan di mode Tanpa kontainer ini. Fungsi Lambda berjalan sebagai proses Linux biasa tanpa isolasi apapun.

Fitur ini tersedia untuk AWS IoT Greengrass Core v1.7 dan yang lebih baru.

Kami menyarankan Anda menjalankan fungsi Lambda dalam kontainer Greengrass kecuali kasus penggunaan Anda mengharuskannya untuk menjalankan tanpa kontainerisasi. Ketika fungsi

Lambda Anda berjalan dalam kontainer Greengrass, Anda dapat menggunakan lampiran sumber daya lokal dan perangkat dan mendapatkan manfaat dari isolasi dan peningkatan keamanan. Sebelum Anda mengubah kontainerisasi, lihat the section called “Pertimbangan ketika memilih fungsi Lambda kontainerisasi”.

> **Note**
>
> Untuk menjalankan tanpa mengaktifkan namespace kernel perangkat dan cgroup, semua fungsi Lambda Anda harus berjalan tanpa kontainerisasi. Anda dapat mudah melakukannya dengan menetapkan kontainerisasi default untuk grup. Untuk informasi, lihat the section called “Pengaturan kontainerisasi default untuk fungsi Lambda dalam grup”.

Batas memori

Alokasi memori untuk fungsi. Ukuran default-nya adalah 16 MB.

> **Note**
>
> Pengaturan batas memori menjadi tidak tersedia ketika Anda mengubah fungsi Lambda untuk menjalankan tanpa kontainerisasi. Fungsi Lambda yang berjalan tanpa kontainerisasi tidak memiliki batas memori. Pengaturan batas memori dibuang ketika Anda mengubah fungsi Lambda atau grup default kontainerisasi pengaturan untuk menjalankan tanpa kontainerisasi.

Batas waktu

Jumlah waktu sebelum fungsi atau permintaan dihentikan. Default-nya adalah 3 detik.

Disematkan

Siklus hidup fungsi Lambda dapat Sesuai permintaan atau berumur panjang. Default-nya sesuai permintaan.

Fungsi Lambda sesuai permintaan dimulai dalam kontainer baru atau digunakan kembali ketika dipanggil. Permintaan untuk fungsi mungkin diproses oleh wadah yang tersedia. Umur panjang —atau disematkan—Fungsi Lambda dimulai secara otomatis setelah AWS IoT Greengrass dimulai dan terus berjalan dalam kontainer sendiri (atau sandbox). Semua permintaan untuk

fungsi diproses oleh kontainer yang sama. Untuk informasi selengkapnya, lihat the section called "Konfigurasi siklus hidup".

Baca akses ke direktori/sys

Apakah fungsi dapat mengakses host dari folder /sys. Gunakan ini ketika fungsi harus membaca informasi perangkat dari /sys. Default-nya adalah SALAH.

> **Note**
>
> Pengaturan ini tidak tersedia ketika Anda menjalankan fungsi Lambda tanpa kontainerisasi. Nilai pengaturan ini dibuang ketika Anda mengubah fungsi Lambda untuk menjalankan tanpa kontainerisasi.

Jenis pengkodean

Jenis encoding yang diharapkan dari muatan input untuk fungsi, baik JSON atau biner. Default-nya adalah JSON.

Support untuk jenis pengkodean biner tersedia mulai AWS IoT Greengrass Perangkat Lunak Core v1.5.0 dan AWS IoT Greengrass Core SDK v1.1.0. Menerima data input biner dapat berguna untuk fungsi yang berinteraksi dengan data perangkat, karena keterbatasan kemampuan perangkat keras dari perangkat sering membuat sulit atau tidak mungkin bagi mereka untuk membangun tipe data JSON.

> **Note**
>
> Lambda yang dapat dieksekusi mendukung jenis pengkodean biner saja, bukan JSON.

Argumen proses

Argumen baris perintah diteruskan ke fungsi Lambda saat dijalankan.

Variabel lingkungan

Pasangan kunci-nilai yang secara dinamis dapat melewati pengaturan untuk fungsi kode dan perpustakaan. Variabel lingkungan lokal bekerja dengan cara yang sama seperti AWS Lambda variabel lingkungan fungsi, namun tersedia di lingkungan core.

Kebijakan akses sumber daya

Daftar hingga 10 Sumber Daya Lokal, Sumber daya rahasia, dan sumber daya machine learning bahwa fungsi Lambda diizinkan untuk mengakses, dan yang sesuai `read-only` atau izin `read-write` ini. Di konsol, sumber daya afiliasi ini tercantum di halaman konfigurasi grup di tab Sumber Daya.

Mode kontainerisasi mempengaruhi bagaimana fungsi Lambda dapat mengakses perangkat lokal dan sumber daya volume dan sumber daya machine learning.

- Fungsi Lambda nonkontainerisasi harus mengakses sumber daya perangkat dan volume lokal secara langsung melalui sistem file pada perangkat core.
- Untuk memungkinkan fungsi Lambda nonkontainerisasi untuk mengakses sumber daya pembelajaran mesin dalam grup Greengrass, Anda harus mengatur pemilik sumber daya dan properti izin mengakses pada sumber daya machine learning. Untuk informasi selengkapnya, lihat the section called “Mengakses sumber daya machine learning”.

Untuk informasi tentang penggunaan AWS IoT Greengrass API untuk menyetel pengaturan konfigurasi khusus grup untuk fungsi Lambda yang ditentukan pengguna, CreateFunctionDefinitionlihat di Referensi API atau di Referensi AWS IoT Greengrass Version 1 create-function-definitionPerintah. AWS CLI Untuk men-deploy fungsi Lambda ke Greengrass core, membuat ve rsi definisi fungsi yang berisi fungsi Anda, membuat versi grup yang mereferensi versi definisi fungsi dan komponen kelompok lainnya, dan kemudian deploy grup.

Menjalankan fungsi Lambda sebagai root

Fitur ini tersedia untuk AWS IoT Greengrass Core v1.7 dan yang lebih baru.

Sebelum Anda dapat menjalankan satu atau lebih fungsi Lambda sebagai root, Anda harus terlebih dahulu memperbarui konfigurasi AWS IoT Greengrass untuk mengaktifkan dukungan. Support untuk menjalankan fungsi Lambda sebagai root secara default tidak aktif. Deployment gagal jika Anda mencoba untuk men-deploy fungsi Lambda dan menjalankannya sebagai root (UID dan GID 0) dan Anda belum memperbarui konfigurasi AWS IoT Greengrass ini. Kesalahan berikut muncul ketika mencatat waktu aktif (*`greengrass_root`*/ggc/var/log/system/runtime.log):

```
lambda(s)
[list of function arns] are configured to run as root while Greengrass is not
 configured to run lambdas with root permissions
```

> **Important**
>
> Kami merekomendasikan Anda menghindari menjalankan fungsi Lambda sebagai root kecuali benar-benar diperlukan. Menjalankan sebagai root meningkatkan risiko berikut:
>
> - Risiko perubahan yang tidak diinginkan, seperti secara tidak sengaja menghapus file kritis.
> - Risiko terhadap data dan perangkat Anda dari individu jahat.
> - Risiko kontainer lolos ketika kontainer Docker berjalan dengan `--net=host` dan `UID=EUID=0`.

Untuk mengizinkan fungsi Lambda dijalankan sebagai root

1. Pada perangkat AWS IoT Greengrass Anda, navigasikan ke folder *greengrass-root*/config ini.

   > **Note**
   >
   > Secara default, *greengrass-root* adalah direktori /greengrass.

2. Edit file config.json untuk menambahkan `"allowFunctionsToRunAsRoot" : "yes"` ke bidang `runtime` ini. Sebagai contoh:

   ```
   {
     "coreThing" : {
       ...
     },
     "runtime" : {
       ...
       "allowFunctionsToRunAsRoot" : "yes"
     },
     ...
   }
   ```

3. Gunakan perintah berikut untuk memulai ulang AWS IoT Greengrass:

   ```
   cd /greengrass/ggc/core
   sudo ./greengrassd restart
   ```

Sekarang Anda dapat melakukan pengaturan ID pengguna dan grup ID (UID/GID) dari fungsi Lambda ke 0 untuk menjalankan fungsi Lambda tersebut sebagai root.

Anda dapat mengubah nilai `"allowFunctionsToRunAsRoot"` ke `"no"` dan mulai ulang AWS IoT Greengrass jika anda ingin melarang fungsi Lambda untuk dijalankan sebagai root.

Pertimbangan ketika memilih fungsi Lambda kontainerisasi

Fitur ini tersedia untuk AWS IoT Greengrass Core v1.7 dan yang lebih baru.

Secara default, fungsi Lambda berjalan di dalam AWS IoT Greengrass kontainer. Kontainer yang menyediakan isolasi antara fungsi Anda dan host, yang menawarkan keamanan lebih untuk kedua host dan fungsi dalam kontainer.

Kami menyarankan Anda menjalankan fungsi Lambda dalam kontainer Greengrass kecuali kasus penggunaan Anda mengharuskannya untuk menjalankan tanpa kontainerisasi. Dengan menjalankan fungsi Lambda Anda dalam kontainer Greengrass, Anda memiliki lebih banyak kontrol untuk membatasi akses ke sumber daya.

Berikut adalah beberapa contoh kasus penggunaan untuk menjalankan tanpa kontainerisasi:

- Anda ingin menjalankan AWS IoT Greengrass pada perangkat yang tidak mendukung mode kontainer (misalnya, karena Anda menggunakan distribusi Linux khusus atau memiliki versi kernel yang terlalu tua).
- Anda ingin menjalankan fungsi Lambda Anda di lingkungan kontainer lain dengan overlayFS sendiri, tetapi menghadapi konflik overlayFS ketika Anda menjalankan dalam kontainer Greengrass.
- Anda memerlukan akses ke sumber daya lokal dengan jalur yang tidak dapat ditentukan pada waktu deployment atau jalur yang dapat berubah setelah deployment, seperti perangkat pluggable.
- Anda memiliki aplikasi warisan yang ditulis sebagai proses dan Anda mengalami masalah ketika menjalankannya sebagai fungsi Lambda terkontainerisasi.

Perbedaan kontainerisasi

Kontainerisasi	Catatan
Kontainer Greengrass	• Semua fitur AWS IoT Greengrass yang tersedia ketika Anda menjalankan fungsi Lambda dalam kontainer Greengrass. • Fungsi Lambda yang berjalan dalam kontainer Greengrass tidak dapat mengakses ke kode yang di-deploy oleh fungsi Lambda lainnya, bahkan jika mereka berjalan dengan ID grup yang sama. Dengan kata lain, fungsi Lambda Anda berjalan dengan isolasi yang lebih besar dari satu sama lain. • Karena fungsi Lambda yang berjalan di AWS IoT Greengrass memiliki semua proses anak dieksekusi dalam kontainer yang sama dengan fungsi Lambda, proses anak dihentikan ketika fungsi Lambda dihentikan.

Tanpa kontainer	• Fitur-fitur berikut tidak tersedia untuk fungsi Lambda nonkontainerisasi:
	  • Batas memori fungsi Lambda.
	  • Sumber daya perangkat dan volume lokal. Anda harus mengakses sumber daya ini pada perangkat core secara langsung daripada mengaksesnya sebagai anggota grup Greengrass.
	• Jika fungsi Lambda nonkontainerisasi Anda mengakses sumber daya machine learning, Anda harus mengidentifikasi pemilik sumber daya dan mengatur izin akses pada sumber daya, bukan pada fungsi Lambda. Hal ini membutuhkan perangkat lunak AWS IoT Greengrass core v1.10 atau yang lebih baru. Untuk informasi selengkapnya, lihat the

Kontainerisasi	Catatan
	section called “Mengakses sumber daya machine learning”. • Fungsi Lambda memiliki akses hanya-baca ke kode yang di-deploy oleh fungsi Lambda lain yang berjalan dengan ID grup yang sama. • Fungsi Lambda yang menghasilkan proses anak dalam sesi proses yang berbeda atau dengan penangan SIGHUP (sinyal hangup) oleh ahli, seperti dengan utilitas nohup, tidak secara otomatis diakhiri oleh AWS IoT Greengrass ketika fungsi Lambda induk diakhiri.

> **Note**
>
> Pengaturan kontainerisasi default untuk grup Greengrass tidak berlaku untuk konektor.

Mengubah kontainerisasi untuk fungsi Lambda dapat menyebabkan masalah ketika Anda men-deploy-nya. Jika Anda telah menetapkan sumber daya lokal untuk fungsi Lambda Anda yang tidak lagi tersedia dengan pengaturan kontainerisasi baru Anda, deployment bisa gagal.

- Ketika Anda mengubah fungsi Lambda dari berjalan dalam kontainer Greengrass menjadi berjalan tanpa kontainerisasi, batas memori untuk fungsi dibuang. Anda harus mengakses sistem file secara langsung daripada menggunakan sumber daya lokal terlampir. Anda harus menghapus sumber daya terlampir sebelum Anda men-deploy.
- Ketika Anda mengubah fungsi Lambda dari berjalan tanpa kontainerisasi menjadi berjalan dalam kontainer, fungsi Lambda Anda kehilangan akses langsung ke sistem file. Anda harus menentukan batas memori untuk setiap fungsi atau menerima default 16 MB. Anda dapat mengonfigurasi pengaturan tersebut untuk setiap fungsi Lambda sebelum Anda men-deploy.

Untuk mengubah pengaturan kontainerisasi pada fungsi Lambda

1. Di panel navigasi AWS IoT konsol, di bawah Kelola, perluas perangkat Greengrass, lalu pilih Grup (V1).
2. Pilih grup yang berisi fungsi Lambda yang pengaturannya ingin Anda ubah.
3. Pilih tab Fungsi Lambda.
4. Di fungsi Lambda yang ingin Anda ubah, pilih elipsis (...) dan kemudian pilih Edit konfigurasi.
5. Mengubah pengaturan kontainerisasi. Jika Anda mengonfigurasi fungsi Lambda untuk menjalankan dalam kontainer Greengrass, Anda juga harus mengatur Batas memori dan akses Baca ke direktori /sys.
6. Pilih Simpan dan kemudian Konfirmasi untuk menyimpan perubahan pada fungsi Lambda Anda.

Perubahan berlaku ketika grup di-deploy.

Anda juga dapat menggunakan CreateFunctionDefinitiondan CreateFunctionDefinitionVersiondi Referensi AWS IoT Greengrass API. Jika Anda mengubah pengaturan kontainerisasi, pastikan untuk memperbarui parameter lainnya juga. Sebagai contoh, jika Anda mengubah dari menjalankan fungsi Lambda dalam kontainer Greengrass menjadi berjalan tanpa kontainerisasi, pastikan untuk menghapus parameter `MemorySize` ini.

Tentukan mode isolasi yang didukung oleh perangkat Greengrass Anda

Anda dapat menggunakan dependensi checker AWS IoT Greengrass untuk menentukan mode isolasi (kontainer Greengrass) yang didukung oleh perangkat Greengrass Anda.

Untuk menjalankan perintah dependency checker AWS IoT Greengrass ini

1. Unduh dan jalankan pemeriksa AWS IoT Greengrass ketergantungan dari repositori. GitHub

```
wget https://github.com/aws-samples/aws-greengrass-samples/raw/master/greengrass-
dependency-checker-GGCv1.11.x.zip
unzip greengrass-dependency-checker-GGCv1.11.x.zip
cd greengrass-dependency-checker-GGCv1.11.x
sudo modprobe configs
sudo ./check_ggc_dependencies | more
```

2. Di mana `more` muncul, tekan tombol kunci Spacebar untuk menampilkan halaman teks lainnya.

Untuk informasi tentang perintah modprobe ini, jalankan man modprobe di terminal.

Mengatur identitas akses default untuk fungsi Lambda dalam grup

Fitur ini tersedia untuk AWS IoT Greengrass Core v1.8 dan yang lebih baru.

Untuk kontrol lebih lanjut atas akses ke sumber daya perangkat, Anda dapat mengonfigurasi identitas akses default yang digunakan untuk menjalankan fungsi Lambda dalam grup. Pengaturan ini menentukan izin default yang diberikan ke fungsi Lambda Anda ketika mereka berjalan pada perangkat core. Untuk mengesampingkan pengaturan untuk fungsi individu dalam grup, Anda dapat menggunakan fungsi Jalankan sebagai properti. Untuk informasi lebih lanjut, lihat Jalankan sebagai.

Pengaturan tingkat kelompok ini juga digunakan untuk menjalankan perangkat lunak Core AWS IoT Greengrass yang mendasarinya. Ini terdiri dari fungsi sistem Lambda yang mengelola operasi, seperti perutean pesan, sinkronisasi bayangan lokal, dan deteksi alamat IP otomatis.

Identitas akses default dapat dikonfigurasi untuk dijalankan sebagai standar AWS IoT Greengrass akun sistem (ggc_user dan ggc_group) atau gunakan izin dari pengguna atau grup lain. Kami merekomendasikan Anda mengonfigurasi perangkat keras Greengrass Anda dengan batas sumber daya, izin file, dan kuota disk yang sesuai untuk setiap pengguna dan grup yang izinnya digunakan untuk menjalankan fungsi Lambda yang ditentukan pengguna atau sistem.

Untuk mengubah identitas akses default untuk grup AWS IoT Greengrass Anda

1. Di panel navigasi AWS IoT konsol, di bawah Kelola, perluas perangkat Greengrass, lalu pilih Grup (V1).
2. Pilih grup yang pengaturannya ingin Anda ubah.
3. Pilih tab fungsi Lambda dan, di bawah bagian lingkungan runtime fungsi Lambda Default, pilih Edit.
4. Di halaman Edit lingkungan runtime fungsi Lambda default, di bawah Pengguna dan grup sistem default, pilih ID pengguna/ID grup lain.

   Saat Anda memilih opsi ini, kolom ID pengguna sistem (nomor) dan ID grup sistem (angka) ditampilkan.
5. Masukkan ID pengguna, ID grup, atau keduanya. Jika Anda meninggalkan bidang kosong, akun sistem Greengrass masing-masing (ggc_user atau ggc_group) digunakan.

   - Untuk ID pengguna Sistem (nomor), masukkan ID pengguna untuk pengguna yang memiliki izin yang ingin Anda gunakan secara default untuk menjalankan fungsi Lambda dalam grup.

Anda dapat menggunakan perintah getent passwd pada perangkat AWS IoT Greengrass Anda untuk mencari ID pengguna.

- Untuk ID grup sistem (angka), masukkan ID grup untuk grup yang memiliki izin yang ingin Anda gunakan secara default untuk menjalankan fungsi Lambda dalam grup. Anda dapat menggunakan getent group perintah pada perangkat AWS IoT Greengrass Anda untuk mencari ID grup.

> **Important**
>
> Berjalan sebagai pengguna root meningkatkan risiko terhadap data dan perangkat Anda. Jangan jalankan sebagai root (UID/GID = 0) kecuali kasus bisnis Anda memerlukannya. Untuk informasi selengkapnya, lihat the section called “Menjalankan fungsi Lambda sebagai root”.

Perubahan berlaku ketika grup di-deploy.

Pengaturan kontainerisasi default untuk fungsi Lambda dalam grup

Fitur ini tersedia untuk AWS IoT Greengrass Core v1.7 dan yang lebih baru.

Pengaturan kontainerisasi untuk grup Greengrass menentukan kontainerisasi default untuk fungsi Lambda dalam grup.

- Dalam mode kontainer Greengrass, fungsi Lambda berjalan di lingkungan waktu aktif terisolasi di dalam AWS IoT Greengrass kontainer secara default.
- Dalam mode Tanpa kontainer ini, fungsi Lambda dijalankan sebagai proses Linux reguler secara default.

Anda dapat mengubah pengaturan grup untuk menentukan kontainerisasi default untuk fungsi Lambda dalam grup. Anda dapat mengganti pengaturan ini untuk satu atau lebih fungsi Lambda dalam grup jika Anda ingin fungsi Lambda untuk menjalankan dengan kontainerisasi berbeda dari default grup. Sebelum Anda mengubah pengaturan kontainerisasi, lihat the section called “Pertimbangan ketika memilih fungsi Lambda kontainerisasi”.

> **Important**
>
> Jika Anda ingin mengubah kontainerisasi default untuk grup, tetapi memiliki satu atau lebih fungsi yang menggunakan kontainerisasi yang berbeda, mengubah pengaturan untuk fungsi Lambda sebelum Anda mengubah pengaturan grup. Jika Anda mengubah pengaturan kontainerisasi grup pertama, nilai untuk Batas memori dan Baca akses ke direktori /sys pengaturan akan dibuang.

Untuk mengubah pengaturan kontainerisasi bagi perangkat Grup AWS IoT Greengrass Anda

1. Di panel navigasi AWS IoT konsol, di bawah Kelola, perluas perangkat Greengrass, lalu pilih Grup (V1).
2. Pilih grup yang pengaturannya ingin Anda ubah.
3. Pilih tab Fungsi Lambda.
4. Di bawah lingkungan runtime fungsi Lambda default, pilih Edit.
5. Di lingkungan runtime fungsi Lambda default Edit, halaman, di bawah kontainerisasi fungsi Lambda Default, ubah setelan containerization.
6. Pilih Simpan.

Perubahan berlaku ketika grup di-deploy.

Arus komunikasi untuk fungsi Greengrass Lambda

Fungsi Greengrass Lambda mendukung beberapa metode komunikasi dengan anggota lain dari grup AWS IoT Greengrass ini, layanan lokal, dan layanan cloud (layanan AWS sudah termasuk).

Komunikasi menggunakan pesan MQTT

fungsi Lambda dapat mengirim dan menerima pesan MQTT menggunakan pola mempublikasikan-berlangganan yang dikendalikan oleh langganan.

Aliran komunikasi ini memungkinkan fungsi Lambda untuk bertukar pesan dengan entitas berikut:

- Perangkat klien dalam grup.
- Konektor dalam grup.

- Fungsi Lambda lainnya dalam grup.
- AWS IoT.
- Layanan Bayangan Perangkat Lokal.

Langganan menentukan sumber pesan, target pesan, dan topik (atau subjek) yang digunakan untuk mengarahkan pesan dari sumber ke target. Pesan yang dipublikasikan ke fungsi Lambda dilewatkan ke handler terdaftar fungsi ini. Langganan memungkinkan lebih banyak keamanan dan memberikan interaksi yang dapat diprediksi. Untuk informasi selengkapnya, lihat the section called “Langganan yang dikelola di dalam alur kerja pesan MQTT”.

> **Note**
>
> Ketika inti sedang offline, fungsi Greengrass Lambda dapat bertukar pesan dengan perangkat klien, konektor, fungsi lain, dan bayangan lokal, tetapi pesan ke antri. AWS IoT Untuk informasi selengkapnya, lihat the section called “Antrean pesan MQTT”.

Arus komunikasi lainnya

- Untuk berinteraksi dengan perangkat lokal dan sumber daya volume dan model machine learning pada perangkat core, fungsi Greengrass Lambda menggunakan antarmuka dari sistem operasi platform spesifik. Misalnya, Anda dapat menggunakan metode `open` pada modul os dalam fungsi Python. Untuk memungkinkan fungsi dapat mengakses sumber daya, fungsi harus berafiliasi dengan sumber daya dan diberikan izin `read-only` atau `read-write` ini. Untuk informasi lebih lanjut, termasuk ketersediaan AWS IoT Greengrass versi core, lihat Akses sumber daya lokal dan the section called “Mengakses sumber daya machine learning dari kode fungsi Lambda”.

  > **Note**
  >
  > Jika Anda menjalankan fungsi Lambda Anda tanpa kontainerisasi, Anda tidak dapat menggunakan perangkat lokal terlampir dan sumber daya volume dan harus mengakses sumber daya tersebut secara langsung.

- Fungsi Lambda dapat menggunakan klien `Lambda` di AWS IoT Greengrass Core SDK untuk memanggil fungsi Lambda lainnya dalam grup Greengrass.
- fungsi Lambda dapat menggunakan AWS SDK untuk berkomunikasi dengan layanan AWS ini. Untuk informasi lebih lanjut, lihat AWS SDK.

- Fungsi Lambda dapat menggunakan antarmuka pihak ketiga untuk berkomunikasi dengan layanan cloud eksternal, mirip dengan fungsi Lambda berbasis cloud.

> **Note**
>
> Fungsi Greengrass Lambda tidak dapat berkomunikasi dengan AWS atau layanan cloud lainnya ketika core sedang offline.

Mengambil input topik MQTT (atau subjek)

AWS IoT Greengrassmenggunakan langganan untuk mengontrol pertukaran pesan MQTT antara perangkat klien, fungsi Lambda, dan konektor dalam grup, dan dengan AWS IoT atau layanan bayangan lokal. Langganan menentukan sumber pesan, target pesan, dan topik MQTT yang digunakan untuk rute pesan. Ketika target adalah fungsi Lambda, handler fungsi dipanggil ketika sumber menerbitkan pesan. Untuk informasi selengkapnya, lihat the section called “Komunikasi menggunakan pesan MQTT”.

Contoh berikut menunjukkan bagaimana fungsi Lambda bisa mendapatkan input topik dari `context` yang diteruskan ke handler. Hal ini dilakukan dengan mengakses kunci `subject` dari hierarki konteks (`context.client_context.custom['subject']`). Contoh ini juga mengurai input pesan JSON dan kemudian menerbitkan topik dan pesan yang diurai.

> **Note**
>
> Di API AWS IoT Greengrass ini, topik langganan diwakili oleh properti `subject` ini.

```
import greengrasssdk
import logging

client = greengrasssdk.client('iot-data')

OUTPUT_TOPIC = 'test/topic_results'

def get_input_topic(context):
    try:
        topic = context.client_context.custom['subject']
    except Exception as e:
```

```
        logging.error('Topic could not be parsed. ' + repr(e))
    return topic

def get_input_message(event):
    try:
        message = event['test-key']
    except Exception as e:
        logging.error('Message could not be parsed. ' + repr(e))
    return message

def function_handler(event, context):
    try:
        input_topic = get_input_topic(context)
        input_message = get_input_message(event)
        response = 'Invoked on topic "%s" with message "%s"' % (input_topic,
 input_message)
        logging.info(response)
    except Exception as e:
        logging.error(e)

    client.publish(topic=OUTPUT_TOPIC, payload=response)

    return
```

Untuk menguji fungsi, tambahkan ke grup Anda menggunakan pengaturan konfigurasi default. Kemudian, tambahkan langganan berikut dan deploy grupnya. Untuk petunjuk, lihat [the section called "Modul 3 (bagian 1): Fungsi Lambda padaAWS IoT Greengrass"](#).

Sumber
topik

Fungsi
t_message

Sumber
topik

Fungsi
c_results

Setelah deployment selesai, mengaktifkan fungsi.

1. Di AWS IoT konsol, buka halaman klien pengujian MQTT.
2. Berlangganan `test/topic_results` topik dengan memilih tab Berlangganan ke topik.
3. Publikasikan pesan ke `test/input_message` topik dengan memilih tab Publikasikan ke topik. Untuk contoh ini, Anda harus menyertakan properti `test-key` di pesan JSON.

   ```
   {
     "test-key": "Some string value"
   }
   ```

   Jika berhasil, fungsi menerbitkan input topik dan string pesan ke topik `test/topic_results` ini.

Konfigurasi siklus hidup untuk fungsi Greengrass Lambda

Siklus hidup fungsi Greengrass Lambda menentukan kapan fungsi dimulai dan bagaimana menciptakan dan menggunakan kontainer. Siklus hidup juga menentukan bagaimana variabel dan logika preprocessing yang berada di luar fungsi handler dipertahankan.

AWS IoT Greengrass dukungan siklus hidup sesuai permintaan (default) atau berumur panjang:

- Fungsi Sesuai permintaan mulai ketika mereka dipanggil dan berhenti ketika tidak ada tugas yang tersisa untuk dieksekusi. Pemanggilan fungsi membuat kontainer (atau sandbox) terpisah untuk memproses pemanggilan, kecuali jika kontainer yang ada dapat digunakan kembali. Data yang dikirim ke fungsi mungkin ditarik oleh salah satu kontainer.

  Beberapa pemanggilan fungsi sesuai permintaan dapat berjalan secara paralel.

  Variabel dan logika preprocessing yang didefinisikan di luar fungsi handler tidak dipertahankan ketika kontainer baru dibuat.

- Fungsi yang Berumur panjang (atau disematkan) mulai secara otomatis ketika core AWS IoT Greengrass dimulai dan berjalan dalam kontainer tunggal. Semua data yang dikirim ke fungsi ditarik oleh kontainer yang sama.

  Beberapa pemanggilan diantrekan hingga pemanggilan sebelumnya dieksekusi.

Variabel dan logika preprocessing yang didefinisikan di luar fungsi handler dipertahankan untuk setiap pemanggilan handler.

Fungsi Lambda yang berumur panjang berguna ketika Anda harus mulai melakukan pekerjaan tanpa input awal. Misalnya, fungsi yang berumur panjang dapat memuat dan mulai memproses model ML agar siap ketika fungsi mulai menerima data perangkat.

> **Note**
>
> Ingat bahwa fungsi yang berumur panjang memiliki timeout yang terkait dengan pemanggilan dari handler-nya. Jika Anda ingin menjalankan kode tanpa batas waktu, Anda harus memulainya di luar handler. Pastikan bahwa tidak ada kode pemblokiran di luar handler yang mungkin mencegah fungsi menyelesaikan inisialisasi.
> Fungsi-fungsi ini berjalan kecuali core berhenti (misalnya, selama deployment grup atau perangkat reboot) atau fungsi memasuki status error (seperti timeout handler, tidak masuk pengecualian, atau ketika fungsi melebihi batas memori).

Untuk informasi lebih lanjut tentang penggunaan kembali kontainer, lihat Memahami Penggunaan Kembali Kontainer di AWS Lambda dalam Blog Komputasi AWS ini.

Lambda yang dapat dieksekusi

Fitur ini tersedia untuk AWS IoT Greengrass Core v1.6 dan yang lebih baru.

Sebuah Lambda yang dapat dieksekusi adalah jenis fungsi Greengrass Lambda yang dapat Anda gunakan untuk menjalankan kode biner di lingkungan core. Ini memungkinkan Anda menjalankan fungsionalitas khusus perangkat secara asli dan mendapat manfaat dari jejak kode kompilasi yang lebih kecil. Lambda yang dapat dieksekusi dapat dipanggil oleh beberapa peristiwa, memanggil fungsi lainnya, dan mengakses sumber daya lokal.

Lambda yang dapat dieksekusi mendukung jenis encoding biner saja (tidak JSON), tetapi jika tidak, anda dapat mengelola mereka dalam kelompok Greengrass Anda dan menyebarkan mereka seperti fungsi Greengrass Lambda lainnya. Namun, proses menciptakan Lambda yang dapat dieksekusi berbeda dari menciptakan Python, Java, dan Node.js Lambda fungsi:

- Anda tidak dapat menggunakan konsol AWS Lambda untuk membuat (atau mengelola) Lambda yang dapat dieksekusi. Anda dapat membuat Lambda yang dapat dieksekusi hanya dengan menggunakan API AWS Lambda ini.
- Anda mengunggah kode fungsi ke AWS Lambda sebagai kompilasi yang dapat dieksekusi yang mencakup AWS IoT Greengrass Core SDK for C.
- Anda menentukan nama yang dapat dieksekusi sebagai fungsi handler.

Lambda yang dapat dieksekusi harus menerapkan panggilan tertentu dan pola pemrograman dalam kode fungsi mereka. Misalnya, metode `main` harus:

- Panggilan `gg_global_init` untuk menginisialisasi variabel global internal Greengrass. Fungsi ini harus dipanggil sebelum membuat thread apapun, dan sebelum memanggil fungsi AWS IoT Greengrass Core SDK yang lain.
- Panggilan `gg_runtime_start` untuk mendaftarkan fungsi handler dengan waktu aktif Greengrass Lambda. Fungsi ini harus dipanggil selama inisialisasi. Memanggil fungsi ini menyebabkan thread digunakan oleh waktu aktif. Opsional parameter `GG_RT_OPT_ASYNC` memberitahu fungsi ini untuk tidak memblokir, melainkan untuk membuat thread baru untuk waktu aktif. Fungsi ini menggunakan handler `SIGTERM` ini.

Cuplikan berikut adalah `main` metode dari contoh kode simple_handler.c pada. GitHub

```
int main() {
    gg_error err = GGE_SUCCESS;

    err = gg_global_init(0);
    if(err) {
        gg_log(GG_LOG_ERROR, "gg_global_init failed %d", err);
        goto cleanup;
    }

    gg_runtime_start(handler, 0);

cleanup:
    return -1;
}
```

Untuk informasi lebih lanjut tentang persyaratan, kendala, dan rincian implementasi lainnya, lihat AWS IoT Greengrass Core SDK for C.

Buat Lambda yang dapat dieksekusi

Setelah Anda mengkompilasi kode Anda bersama dengan SDK, gunakan API AWS Lambda untuk membuat fungsi Lambda dan mengunggah kompilasi yang dapat dieksekusi milik Anda.

> **Note**
>
> Fungsi Anda harus dikompilasi dengan compiler C89 yang kompatibel.

Contoh berikut menggunakan perintah CLI buat fungsi untuk membuat Lambda yang dapat dieksekusi. Perintah menentukan:

- Nama eksekusi untuk handler. Hal ini harus menjadi nama yang tepat dari kompilasi yang dapat dieksekusi milik Anda.
- Jalur ke file `.zip` yang berisi kompilasi yang dapat dieksekusi.
- `arn:aws:greengrass:::runtime/function/executable` untuk waktu aktif. Ini adalah waktu aktif untuk semua Lambda yang dapat dieksekusi.

> **Note**
>
> Untuk `role`, Anda dapat menentukan ARN dari setiap peran eksekusi Lambda. AWS IoT Greengrass tidak menggunakan peran ini, tetapi parameter diperlukan untuk membuat fungsi. Untuk informasi lebih lanjut tentang peran eksekusi Lambda, lihat AWS Lambda model izin dalam AWS Lambda Panduan Developer.

```
aws lambda create-function \
--region aws-region \
--function-name function-name \
--handler executable-name \
--role role-arn \
--zip-file fileb://file-name.zip \
--runtime arn:aws:greengrass:::runtime/function/executable
```

Selanjutnya, gunakan API AWS Lambda untuk menerbitkan versi dan membuat alias.

- Gunakan publikasikan versi untuk mempublikasikan versi fungsi.

```
aws lambda publish-version \
--function-name function-name \
--region aws-region
```

- Gunakan Buat alias alias untuk membuat alias poin ke versi yang baru saja Anda publikasikan. Kami menyarankan Anda mereferensi fungsi Lambda dengan alias ketika Anda menambahkannya ke grup Greengrass.

```
aws lambda create-alias \
--function-name function-name \
--name alias-name \
--function-version version-number \
--region aws-region
```

> **Note**
>
> Konsol AWS Lambda tidak menampilkan Lambda yang dapat dieksekusi. Untuk memperbarui kode fungsi, Anda harus menggunakan API AWS Lambda ini.

Kemudian, menambahkan Lambda yang dapat dieksekusi untuk grup Greengrass, mengonfigurasinya untuk menerima data input biner dalam pengaturan grup tertentu, dan men-deploy grup. Anda dapat melakukannya di konsol AWS IoT Greengrass tersebut atau menggunakan API AWS IoT Greengrass ini.

Menjalankan AWS IoT Greengrass di kontainer Docker

AWS IoT Greengrass dapat dikonfigurasi untuk berjalan di kontainer Docker ini.

Anda dapat mengunduh Dockerfile melalui AmazonCloudFront yang memiliki perangkat lunakAWS IoT Greengrass Core dan dependensi yang terinstal. Untuk mengubah citra Docker agar berjalan pada arsitektur platform yang berbeda atau mengurangi ukuran citra Docker, lihat file `README` dalam unduhan paket Docker.

Untuk membantu Anda memulai bereksperimen dengan AWS IoT Greengrass, AWS juga menyediakan prebuilt citra Docker yang memiliki perangkat lunak AWS IoT Greengrass Core dan dependensi yang terinstal. Anda dapat mengunduh citra dari Hub Docker atau Amazon Elastic

Container Registry (Amazon ECR). Citra prebuilt ini menggunakan citra dasar Amazon Linux 2 (x86_64) dan Alpine Linux (x86_64, Armv7l, atau AArch64).

> **Important**
>
> 30 Juni 2022, pemeliharaanAWS IoT Greengrass berakhir untuk perangkat lunakAWS IoT Greengrass Core v1.x Citra Docker yang dipublikasikan ke Amazon Elastic Container Registry (Amazon ECR) dan Docker Hub. Anda dapat terus mengunduh gambar Docker ini dari Amazon ECR dan Docker Hub hingga 30 Juni 2023, yaitu 1 tahun setelah pemeliharaan berakhir. Namun, perangkat lunakAWS IoT Greengrass Core v1.x gambar Docker tidak lagi menerima patch keamanan atau perbaikan bug setelah pemeliharaan berakhir pada 30 Juni 2022. Jika Anda menjalankan beban kerja produksi yang bergantung pada image Docker ini, sebaiknya buat image Docker sendiri menggunakan Dockerfiles yangAWS IoT Greengrass menyediakan. Untuk informasi selengkapnya, lihat AWS IoT Greengrass Perangkat lunak Docker.

Topik ini menjelaskan cara mengunduh citra Docker AWS IoT Greengrass dari Amazon ECR dan menjalankannya pada platform Windows, MacOS, atau Linux (x86_64). Topik ini mengandung langkah-langkah berikut:

1. Dapatkan citra kontainer AWS IoT Greengrass dari Amazon ECR
2. Membuat dan mengonfigurasi grup Greengrass dan core
3. Jalankan AWS IoT Greengrass secara lokal
4. Mengonfigurasi kontainerisasi "Tanpa kontainer" untuk grup
5. Men-deploy fungsi Lambda ke kontainer Docker
6. (Opsional) Men-deploy perangkat klien yang berinteraksi dengan Greengrass berjalan dalam kontainer Docker

Fitur-fitur berikut tidak didukung ketika Anda menjalankan AWS IoT Greengrass dalam kontainer Docker:

- Konektor yang berjalan di mode kontainer Greengrass ini. Untuk menjalankan konektor dalam kontainer Docker, konektor harus berjalan dengan mode Tanpa kontainer ini. Untuk menemukan konektor yang mendukung mode Tanpa kontainer ini, lihat the section called “AWS-disediakan

konektor Greengrass". Beberapa konektor ini memiliki parameter mode isolasi yang harus Anda atur ke Tanpa kontainer.

- Sumber daya perangkat dan volume lokal. Fungsi Lambda yang ditetapkan pengguna milik Anda yang berjalan dalam kontainer Docker harus mengakses perangkat dan volume pada core secara langsung.

Fitur-fitur ini tidak didukung ketika lingkungan waktu aktif Lambda untuk grup Greengrass diatur ke Tanpa kontainer, yang diperlukan untuk menjalankan AWS IoT Greengrass dalam kontainer Docker.

Prasyarat

Sebelum Anda memulai tutorial ini, Anda harus melakukan hal berikut.

- Anda harus menginstal perangkat lunak dan versi berikut pada komputer host Anda berdasarkan versi AWS Command Line Interface (AWS CLI) yang Anda pilih.

  AWS CLI version 2

  - Docker versi 18.09 atau yang lebih baru. Versi sebelumnya juga dapat berfungsi, namun kami merekomendasikan 18.09 atau yang lebih baru.
  - AWS CLI versi 2.0.0 atau yang lebih baru.
    - Untuk menginstal AWS CLI versi 2, lihat Menginstal AWS CLI versi 2.
    - Untuk mengonfigurasi AWS CLI, lihat Mengonfigurasi AWS CLI.

    > **Note**
    >
    > Untuk meningkatkan AWS CLI versi 2 ke yang lebih baru pada komputer Windows, Anda harus mengulangi proses instalasi MSI ini.

  AWS CLI version 1

  - Docker versi 18.09 atau yang lebih baru. Versi sebelumnya juga dapat berfungsi, namun kami merekomendasikan 18.09 atau yang lebih baru.
  - Python versi 3.6 atau lebih baru.
  - pip versi 18.1 atau yang lebih baru.
  - AWS CLI versi 1.17.10 atau yang lebih baru
    - Untuk menginstal AWS CLI versi 1, lihat Menginstal AWS CLI versi 1.
    - Untuk mengonfigurasi AWS CLI, lihat Mengonfigurasi AWS CLI.

- Untuk meningkatkan AWS CLI versi 1 ke versi terbaru, jalankan perintah berikut.

```
pip install awscli --upgrade --user
```

> **Note**
>
> Jika Anda menggunakan Instalasi MSI dari AWS CLI versi 1 pada Windows, perhatikan hal-hal berikut:
>
> - Jika instalasi AWS CLI versi 1 gagal untuk menginstal botocore, coba gunakan Instalasi Python dan pip.
> - Untuk meningkatkan AWS CLI versi 1 ke yang lebih baru, anda mesti mengulangi proses instalasi MSI.

- Untuk mengakses sumber daya Amazon Elastic Container Registry (Amazon ECR), Anda harus memberikan izin berikut.
  - Amazon ECR mengharuskan pengguna memiliki izin `ecr:GetAuthorizationToken` melalui kebijakan (IAM) AWS Identity and Access Management sebelum mereka dapat mengautentikasi ke registrasi dan mendorong atau menarik citra dari repositori Amazon ECR. Untuk informasi lebih lanjut, lihat Contoh Kebijakan repositori Amazon ECR dan Mengakses Satu Repositori Amazon ECR dalam Panduan Pengguna Amazon Elastis Registri Kontainer.

Langkah 1: Dapatkan citra kontainer AWS IoT Greengrass dari Amazon ECR

AWS menyediakan citra Docker yang memiliki perangkat lunak AWS IoT Greengrass core terpasang.

> **Warning**
>
> Dimulai dengan v1.11.6 perangkat lunakAWS IoT Greengrass Core, gambar Greengrass Docker tidak lagi menyertakan Python 2.7, karena Python 2.7 mencapai end-of-life pada tahun 2020 dan tidak lagi menerima pembaruan keamanan. Jika Anda memilih untuk memperbarui ke image Docker ini, kami sarankan Anda memvalidasi bahwa aplikasi Anda bekerja dengan image Docker baru sebelum Anda menerapkan pembaruan ke perangkat produksi. Jika Anda memerlukan Python 2.7 untuk aplikasi Anda yang menggunakan gambar

Greengrass Docker, Anda dapat memodifikasi Greengrass Dockerfile untuk menyertakan Python 2.7 untuk aplikasi Anda.

Untuk langkah-langkah yang menunjukkan cara menarik`latest` gambar dari Amazon ECR, pilih sistem operasi Anda:

Tarik citra kontainer (Linux)

Jalankan perintah berikut di terminal komputer Anda.

1. Masuk ke registri AWS IoT Greengrass dalam Amazon ECR.

   ```
   aws ecr get-login-password --region  us-west-2 | docker login --username AWS --
   password-stdin https://216483018798.dkr.ecr.us-west-2.amazonaws.com
   ```

   Jika berhasil, output akan mencetak `Login Succeeded`.

2. Mengambil bagian citra kontainer AWS IoT Greengrass ini.

   ```
   docker pull 216483018798.dkr.ecr.us-west-2.amazonaws.com/aws-iot-greengrass:latest
   ```

   > **Note**
   >
   > Citra `latest` berisi versi stabil terbaru dari Perangkat lunak AWS IoT Greengrass Core yang terinstal pada gambar dasar Amazon Linux 2. Anda juga dapat menarik gambar lain dari repositori. Untuk menemukan semua gambar yang tersedia, periksa Tag pada halaman Docker Hub atau menggunakan perintah aws ecr list-images ini. Misalnya:
   >
   > ```
   > aws ecr list-images --region us-west-2 --registry-id 216483018798 --
   > repository-name aws-iot-greengrass
   > ```

3. Mengaktifkan perlindungan symlink dan hardlink. Jika Anda bereksperimen dengan menjalankan AWS IoT Greengrass dalam sebuah kontainer, Anda dapat mengaktifkan pengaturan untuk boot ketika ini saja.

> **Note**
>
> Anda mungkin harus menggunakan sudo untuk menjalankan perintah ini.

- Untuk mengaktifkan pengaturan untuk boot ketika ini saja:

```
echo 1 > /proc/sys/fs/protected_hardlinks
echo 1 > /proc/sys/fs/protected_symlinks
```

- Untuk mengaktifkan pengaturan untuk bertahan di restart:

```
echo '# AWS IoT Greengrass' >> /etc/sysctl.conf
echo 'fs.protected_hardlinks = 1' >> /etc/sysctl.conf
echo 'fs.protected_symlinks = 1' >> /etc/sysctl.conf

sysctl -p
```

4. Mengaktifkan penerusan jaringan IPv4, yang diperlukan untuk cloud deployment AWS IoT Greengrass dan komunikasi MQTT agar bekerja di Linux. Di file `/etc/sysctl.conf` ini, atur `net.ipv4.ip_forward` ke 1, dan kemudian muat ulang `sysctls`.

```
sudo nano /etc/sysctl.conf
# set this net.ipv4.ip_forward = 1
sudo sysctl -p
```

> **Note**
>
> Anda bisa menggunakan editor pilihan Anda dan bukan nano.

Tarik citra kontainer (macOS)

Jalankan perintah berikut di terminal komputer Anda.

1. Masuk ke registri AWS IoT Greengrass dalam Amazon ECR.

```
aws ecr get-login-password --region  us-west-2 | docker login --username AWS --
password-stdin https://216483018798.dkr.ecr.us-west-2.amazonaws.com
```

Jika berhasil, output akan mencetak `Login Succeeded`.

2. Mengambil bagian citra kontainer AWS IoT Greengrass ini.

```
docker pull 216483018798.dkr.ecr.us-west-2.amazonaws.com/aws-iot-greengrass:latest
```

> **Note**
>
> Citra `latest` berisi versi stabil terbaru dari Perangkat lunak AWS IoT Greengrass Core yang terinstal pada gambar dasar Amazon Linux 2. Anda juga dapat menarik gambar lain dari repositori. Untuk menemukan semua gambar yang tersedia, periksa Tag pada halaman Docker Hub atau menggunakan perintah aws ecr list-images ini. Misalnya:
>
> ```
> aws ecr list-images --region us-west-2 --registry-id 216483018798 --
> repository-name aws-iot-greengrass
> ```

Tarik citra kontainer (Windows)

Jalankan perintah berikut di prompt perintah. Sebelum Anda dapat menggunakan perintah Docker pada Windows, Docker Desktop harus berjalan.

1. Masuk ke registri AWS IoT Greengrass dalam Amazon ECR.

```
aws ecr get-login-password --region  us-west-2 | docker login --username AWS --
password-stdin https://216483018798.dkr.ecr.us-west-2.amazonaws.com
```

Jika berhasil, output akan mencetak `Login Succeeded`.

2. Mengambil bagian citra kontainer AWS IoT Greengrass ini.

```
docker pull 216483018798.dkr.ecr.us-west-2.amazonaws.com/aws-iot-greengrass:latest
```

> **Note**
>
> Citra `latest` berisi versi stabil terbaru dari Perangkat lunak AWS IoT Greengrass Core yang terinstal pada gambar dasar Amazon Linux 2. Anda juga dapat menarik gambar

lain dari repositori. Untuk menemukan semua gambar yang tersedia, periksa Tag pada halaman Docker Hub atau menggunakan perintah aws ecr list-images ini. Misalnya:

```
aws ecr list-images --region us-west-2 --registry-id 216483018798 --
repository-name aws-iot-greengrass
```

Langkah 2: Membuat dan mengonfigurasi grup Greengrass dan core

Citra Docker memiliki perangkat lunak AWS IoT Greengrass core terinstal, tetapi Anda harus membuat grup Greengrass dan core. Ini termasuk mengunduh sertifikat dan file konfigurasi core.

- Ikuti langkah-langkah dalam the section called “Modul 2: MenginstalAWS IoT GreengrassPerangkat lunak inti”. Lewati langkah-langkah di mana Anda mengunduh dan menjalankan perangkat lunakAWS IoT Greengrass Core. Perangkat lunak dan dependensi waktu aktif yang sudah diatur dalam citra Docker.

Langkah 3: Jalankan AWS IoT Greengrass secara lokal

Setelah grup dikonfigurasi, Anda siap untuk mengonfigurasi dan memulai core. Untuk langkah-langkah yang menunjukkan cara melakukannya, pilih sistem operasi Anda:

Jalankan Greengrass secara lokal (Linux)

Jalankan perintah berikut di terminal komputer Anda.

1. Buat folder untuk sumber daya keamanan perangkat, lalu pindahkan sertifikat dan kunci ke folder itu. Jalankan perintah berikut. Ganti *path-to-security-files*dengan jalur ke sumber daya keamanan, dan ganti *certificateId* dengan ID sertifikat dalam nama file.

   ```
   mkdir /tmp/certs
   mv path-to-security-files/certificateId-certificate.pem.crt /tmp/certs
   mv path-to-security-files/certificateId-public.pem.key /tmp/certs
   mv path-to-security-files/certificateId-private.pem.key /tmp/certs
   mv path-to-security-files/AmazonRootCA1.pem /tmp/certs
   ```

2. Buat folder untuk konfigurasi perangkat, dan pindahkan file konfigurasiAWS IoT Greengrass Core ke folder itu. Jalankan perintah berikut. Ganti *path-to-config-file*dengan path ke file konfigurasi.

```
mkdir /tmp/config
mv path-to-config-file/config.json /tmp/config
```

3. Mulai AWS IoT Greengrass dan ikat-pasang sertifikat dan file konfigurasi dalam kontainer Docker.

   Ganti `/tmp` dengan jalur di mana Anda dekompresi sertifikat dan file konfigurasi.

   ```
   docker run --rm --init -it --name aws-iot-greengrass \
   --entrypoint /greengrass-entrypoint.sh \
   -v /tmp/certs:/greengrass/certs \
   -v /tmp/config:/greengrass/config \
   -p 8883:8883 \
   216483018798.dkr.ecr.us-west-2.amazonaws.com/aws-iot-greengrass:latest
   ```

   Output-nya akan terlihat seperti contoh ini:

   ```
   Setting up greengrass daemon
   Validating hardlink/softlink protection
   Waiting for up to 30s for Daemon to start

   Greengrass successfully started with PID: 10
   ```

Jalankan Greengrass secara lokal (macOS)

Jalankan perintah berikut di terminal komputer Anda.

1. Buat folder untuk sumber daya keamanan perangkat, lalu pindahkan sertifikat dan kunci ke folder itu. Jalankan perintah berikut. Ganti *path-to-security-files*dengan jalur ke sumber daya keamanan, dan ganti *certificateId* dengan ID sertifikat dalam nama file.

   ```
   mkdir /tmp/certs
   mv path-to-security-files/certificateId-certificate.pem.crt /tmp/certs
   mv path-to-security-files/certificateId-public.pem.key /tmp/certs
   mv path-to-security-files/certificateId-private.pem.key /tmp/certs
   mv path-to-security-files/AmazonRootCA1.pem /tmp/certs
   ```

2. Buat folder untuk konfigurasi perangkat, dan pindahkan file konfigurasiAWS IoT Greengrass Core ke folder itu. Jalankan perintah berikut. Ganti *path-to-config-file*dengan path ke file konfigurasi.

```
mkdir /tmp/config
mv path-to-config-file/config.json /tmp/config
```

3. Mulai AWS IoT Greengrass dan ikat-pasang sertifikat dan file konfigurasi dalam kontainer Docker.

   Ganti `/tmp` dengan jalur di mana Anda dekompresi sertifikat dan file konfigurasi.

```
docker run --rm --init -it --name aws-iot-greengrass \
--entrypoint /greengrass-entrypoint.sh \
-v /tmp/certs:/greengrass/certs \
-v /tmp/config:/greengrass/config \
-p 8883:8883 \
216483018798.dkr.ecr.us-west-2.amazonaws.com/aws-iot-greengrass:latest
```

   Output-nya akan terlihat seperti contoh ini:

```
Setting up greengrass daemon
Validating hardlink/softlink protection
Waiting for up to 30s for Daemon to start

Greengrass successfully started with PID: 10
```

Jalankan Greengrass drcsrs lokal (Windows)

1. Buat folder untuk sumber daya keamanan perangkat, lalu pindahkan sertifikat dan kunci ke folder itu. Jalankan perintah berikut di prompt perintah. Ganti *path-to-security-files*dengan jalur ke sumber daya keamanan, dan ganti *certificateId* dengan ID sertifikat dalam nama file.

```
mkdir C:\Users\%USERNAME%\Downloads\certs
move path-to-security-files\certificateId-certificate.pem.crt C:\Users\%USERNAME%
\Downloads\certs
move path-to-security-files\certificateId-public.pem.key C:\Users\%USERNAME%
\Downloads\certs
```

```
move path-to-security-files\certificateId-private.pem.key C:\Users\%USERNAME%
\Downloads\certs
move path-to-security-files\AmazonRootCA1.pem C:\Users\%USERNAME%\Downloads\certs
```

2. Buat folder untuk konfigurasi perangkat, dan pindahkan file konfigurasiAWS IoT Greengrass Core ke folder itu. Jalankan perintah berikut di prompt perintah. Ganti *path-to-config-file*dengan path ke file konfigurasi.

```
mkdir C:\Users\%USERNAME%\Downloads\config
move path-to-config-file\config.json C:\Users\%USERNAME%\Downloads\config
```

3. Mulai AWS IoT Greengrass dan ikat-pasang sertifikat dan file konfigurasi dalam kontainer Docker. Jalankan perintah berikut di prompt perintah Anda.

```
docker run --rm --init -it --name aws-iot-greengrass --entrypoint /greengrass-
entrypoint.sh -v c:/Users/%USERNAME%/Downloads/certs:/greengrass/certs
 -v c:/Users/%USERNAME%/Downloads/config:/greengrass/config -p 8883:8883
 216483018798.dkr.ecr.us-west-2.amazonaws.com/aws-iot-greengrass:latest
```

   Saat Docker meminta Anda untuk membagikan drive `C:\` dengan daemon Docker, izinkan drive untuk ikat-pasang `C:\` di dalam kontainer Docker. Untuk informasi lebih lanjut, lihat drive berbagi dalam dokumentasi Docker.

   Output-nya akan terlihat seperti contoh ini:

```
Setting up greengrass daemon
Validating hardlink/softlink protection
Waiting for up to 30s for Daemon to start

Greengrass successfully started with PID: 10
```

> **Note**
>
> Jika kontainer tidak membuka shell dan langsung keluar, Anda dapat men-debug masalah dengan ikat-pasang mencatat waktu aktif Greengrass ketika Anda memulai citra. Untuk informasi selengkapnya, lihat the section called “Untuk dapat bertahan dengan mencatat waktu aktif Greengrass di luar kontainer Docker”.

Langkah 4: Mengonfigurasi kontainerisasi "Tanpa kontainer" untuk grup Greengrass

Ketika Anda menjalankan AWS IoT Greengrass dalam kontainer Docker, semua fungsi Lambda harus berjalan tanpa kontainerisasi. Pada langkah ini, Anda mengatur kontainerisasi default untuk grup Tanpa kontainer. Anda harus melakukan ini sebelum Anda men-deploy grup untuk pertama kalinya.

1. Di panel navigasiAWS IoT konsol, di bawah Kelola, luaskan perangkat Greengrass, lalu pilih Grup (V1).
2. Pilih grup yang pengaturannya ingin Anda ubah.
3. Pilih tab fungsi Lambda.
4. Di bawah Lingkungan runtime fungsi Lambda Default, pilih Edit.
5. Di lingkungan runtime fungsi Lambda Edit default, di bawah Containerization fungsi Lambda Default, ubah pengaturan containerization.
6. Pilih Save (Simpan).

Perubahan berlaku ketika grup di-deploy.

Untuk informasi selengkapnya, lihat the section called “Pengaturan kontainerisasi default untuk fungsi Lambda dalam grup”.

> **Note**
>
> Secara default, fungsi Lambda menggunakan pengaturan grup kontainerisasi. Jika anda mengesampingkan pengaturan Tanpa kontainer untuk setiap fungsi Lambda ketika AWS IoT Greengrass berjalan dalam kontainer Docker, deployment-nya gagal.

Langkah 5: Men-deploy fungsi Lambda ke kontainer Docker AWS IoT Greengrass ini

Anda dapat men-deploy fungsi Lambda berumur panjang ke kontainer Greengrass Docker.

- Ikuti langkah-langkah dalam the section called “Modul 3 (bagian 1): Fungsi Lambda padaAWS IoT Greengrass” untuk men-deploy fungsi Hello World Lambda berumur panjang ke kontainer.

Langkah 6: (Opsional) Men-deploy perangkat klien yang berinteraksi dengan Greengrass berjalan dalam kontainer Docker

Anda juga dapat men-deploy perangkat klien yang berinteraksi denganAWS IoT Greengrass ketika berjalan dalam kontainer Docker.

- Ikuti langkah-langkah dalamthe section called “Modul 4: Berinteraksilah dengan perangkat klien dalamAWS IoT Greengrasskelompok” untuk men-deploy perangkat klien yang connect ke core dan mengirim pesan MQTT.

Menghentikan kontainer Docker AWS IoT Greengrass ini

Untuk menghentikan kontainer Docker AWS IoT Greengrass ini, tekan Ctrl+C di terminal atau prompt perintah. Tindakan ini mengirimkan `SIGTERM` ke proses daemon Greengrass untuk meruntuhkan proses daemon Greengrass dan semua proses Lambda yang dimulai oleh proses daemon. Kontainer Docker diinisialisasi dengan proses `/dev/init` sebagai PID 1, yang membantu untuk menghapus proses zombie sisa. Untuk informasi lebih lanjut, lihat referensi menjalankan Docker.

Penyelesaian masalah AWS IoT Greengrass di kontainer Docker

Gunakan informasi berikut untuk membantu memecahkan masalah dengan menjalankan AWS IoT Greengrass dalam kontainer Docker.

Error: Tidak dapat melakukan login interaktif dari perengakat non TTY.

Solusi: Kesalahan ini dapat terjadi ketika Anda menjalankan perintah `aws ecr get-login-password` ini. Pastikan Anda menginstal AWS CLI versi 2 atau versi 1 yang terbaru. Kami menyarankan agar Anda menggunakan AWS CLI versi 2. Untuk informasi selengkapnya, lihat Menginstal AWS CLI dalam AWS Command Line Interface Panduan Pengguna.

Kesalahan: Opsi tidak diketahui: -no-include-email.

Solusi: Kesalahan ini dapat terjadi ketika Anda menjalankan perintah `aws ecr get-login` ini. Pastikan Anda memiliki versi AWS CLI terbaru yang terinstal (misalnya, jalankan: `pip install awscli --upgrade --user`). Jika Anda menggunakan Windows dan Anda menginstal CLI menggunakan MSI installer, Anda harus mengulangi proses instalasi. Untuk informasi lebih lanjut, lihat Menginstal AWS Command Line Interface di Microsoft Windows dalam AWS Command Line Interface Panduan Pengguna.

Peringatan: IPv4 dinonaktifkan. Jaringan tidak akan bekerja.

Solusi: Anda mungkin menerima peringatan ini atau pesan serupa ketika menjalankan AWS IoT Greengrass pada komputer Linux. Aktifkan penerusan jaringan IPv4 seperti yang dijelaskan dalam Langkah. AWS IoT Greengrass cloud deployment dan komunikasi MQTT tidak bekerja ketika penerusan IPv4 tidak diaktifkan. Untuk informasi lebih lanjut, lihat Mengonfigurasi parameter kernel namespace (sysctls) pada ketika waktu aktif dalam dokumentasi Docker.

Error: Firewall memblokir file berbagi antara windows dan kontainer.

Solusi: Anda mungkin menerima error ini atau pesan `Firewall Detected` ketika menjalankan Docker di komputer Windows. Hal ini juga dapat terjadi jika Anda masuk pada jaringan pribadi virtual (VPN) dan pengaturan jaringan Anda mencegah drive berbagi untuk dipasang. Dalam situasi itu, matikan VPN dan jalankan kembali kontainer Docker.

Kesalahan: Terjadi kesalahan (AccessDeniedException) saat memanggilGetAuthorizationToken operasi: Pengguna: arn:aws:iam: :::user/ <account-id><user-name>tidak berwenang untuk melakukan: ecr: padaGetAuthorizationToken sumber daya: *

Anda mungkin menerima kesalahan ini saat menjalankan `aws ecr get-login-password` jika Anda tidak memiliki izin yang memadai untuk mengakses repositori Amazon ECR. Untuk informasi lebih lanjut, lihat Contoh Kebijakan repositori Amazon ECR dan Mengakses Satu Repositori Amazon ECR dalam Panduan Pengguna Amazon ECR.

Untuk bantuan penyelesaian masalah AWS IoT Greengrass yang umum, lihat Memecahkan masalah.

Debugging AWS IoT Greengrass di kontainer Docker

Untuk debug masalah dengan kontainer Docker, Anda dapat bertahan dengan mencatat waktu aktif Greengrass atau melampirkan shell interaktif untuk kontainer Docker.

Untuk dapat bertahan dengan mencatat waktu aktif Greengrass di luar kontainer Docker

Anda dapat menjalankan kontainer Docker AWS IoT Greengrass setelah ikat-pasang direktori `/greengrass/ggc/var/log` ini. Catatan bertahan bahkan setelah kontainer keluar atau dihapus.

Linux atau macOS

1. Menghentikan kontainer Greengrass Docker berjalan pada host, dan kemudian menjalankan perintah berikut di terminal. Ini ikat-pasang Greengrass direktori `log` dan mulai citra Docker.

Ganti `/tmp` dengan jalur di mana Anda dekompresi sertifikat dan file konfigurasi.

```
docker run --rm --init -it --name aws-iot-greengrass \
      --entrypoint /greengrass-entrypoint.sh \
      -v /tmp/certs:/greengrass/certs \
      -v /tmp/config:/greengrass/config \
      -v /tmp/log:/greengrass/ggc/var/log \
      -p 8883:8883 \
      216483018798.dkr.ecr.us-west-2.amazonaws.com/aws-iot-greengrass:latest
```

Anda kemudian dapat memeriksa log Anda di `/tmp/log` pada host Anda untuk melihat apa yang terjadi ketika Greengrass berjalan di dalam kontainer Docker.

Di Windows

Menghentikan kontainer Greengrass Docker berjalan di host, dan kemudian menjalankan perintah berikut di perintah prompt. Ini ikat-pasang Greengrass direktori `log` dan mulai citra Docker.

```
cd C:\Users\%USERNAME%\Downloads
mkdir log
docker run --rm --init -it --name aws-iot-greengrass --entrypoint /greengrass-
entrypoint.sh -v c:/Users/%USERNAME%/Downloads/certs:/greengrass/certs -v c:/
Users/%USERNAME%/Downloads/config:/greengrass/config -v c:/Users/%USERNAME%/
Downloads/log:/greengrass/ggc/var/log -p 8883:8883 216483018798.dkr.ecr.us-
west-2.amazonaws.com/aws-iot-greengrass:latest
```

Anda kemudian dapat memeriksa log Anda di `C:/Users/%USERNAME%/Downloads/log` pada host Anda untuk melihat apa yang terjadi ketika Greengrass berjalan di dalam kontainer Docker.

Untuk melampirkan shell interaktif ke kontainer Docker

Anda dapat melampirkan shell interaktif untuk menjalankan kontainer Docker AWS IoT Greengrass ini. Hal ini dapat membantu Anda menyelidiki keadaan kontainer Greengrass Docker.

Linux atau macOS

Sementara kontainer Greengrass Docker berjalan, jalankan perintah berikut di terminal terpisah.

```
docker exec -it $(docker ps -a -q -f "name=aws-iot-greengrass") /bin/bash
```

Di Windows

Sementara kontainer Greengrass Docker berjalan, jalankan perintah berikut di prompt perintah terpisah.

```
docker ps -a -q -f "name=aws-iot-greengrass"
```

Ganti *gg-container-id*dengan`container_id` hasil dari perintah sebelumnya.

```
docker exec -it gg-container-id /bin/bash
```

Akses sumber daya lokal dengan fungsi dan konektor Lambda

Fitur ini tersedia untuk AWS IoT Greengrass Core v1.3 dan yang lebih baru.

Dengan AWS IoT Greengrass, Anda dapat fungsi AWS Lambda penulis dan mengonfigurasi konektor di cloud dan men-deploy mereka ke perangkat core untuk eksekusi lokal. Pada core Greengrass yang menjalankan Linux, fungsi Lambda yang men-deploy secara lokal dan konektor dapat mengakses sumber daya lokal yang secara fisik hadir pada perangkat core Greengrass. Sebagai contoh, untuk berkomunikasi dengan perangkat yang terhubung melalui Modbus atau CanBus, Anda dapat mengaktifkan fungsi Lambda Anda untuk mengakses port serial pada perangkat core. Untuk mengonfigurasi akses aman ke sumber daya lokal, Anda harus menjamin keamanan perangkat keras fisik Anda dan core Greengrass perangkat OS Anda.

Untuk mulai mengakses sumber daya lokal, lihat tutorial berikut:

- Cara mengonfigurasi akses sumber daya lokal menggunakan AWS antarmuka baris perintah
- Cara mengonfigurasi akses sumber daya lokal menggunakan AWS Management Console

Jenis sumber daya yang mendukung

Anda dapat mengakses dua jenis sumber daya lokal: sumber daya volume dan sumber daya perangkat.

Sumber daya volume

File atau direktori pada sistem root file (kecuali di bawah `/sys`, `/dev`, atau `/var`). Ini termasuk:

- Folder atau file yang digunakan untuk membaca atau menulis informasi di seluruh fungsi Lambda Greengrass (sebagai contoh, `/usr/lib/python2.x/site-packages/local`).
- Folder atau file di bawah sistem file/proc host (misalnya, `/proc/net` atau `/proc/stat`). Didukung di v1.6 atau versi lebih baru. Untuk keperluan tambahan, lihat the section called “Sumber daya volume di bawah direktori /proc”.

> **Tip**
>
> Untuk mengonfigurasi `/var`, `/var/run`, dan `/var/lib` direktori sebagai sumber volume, pertama-tama pasang direktori di folder lain dan kemudian konfigurasikan folder sebagai sumber daya volume.

Bila Anda mengonfigurasi sumber daya volume, tentukan jalur sumber dan jalur tujuan Anda. Path sumber adalah path absolut dari sumber daya pada host. Jalur tujuan adalah path absolut dari sumber daya di dalam lingkungan namespace Lambda. Ini adalah kontainer yang fungsi Lambda Greengrass atau konektor berjalan di. Setiap perubahan ke jalur tujuan tercermin di jalur sumber pada sistem file host.

> **Note**
>
> File di jalur tujuan terlihat di namespace Lambda saja. Anda tidak dapat melihatnya di namespace Linux biasa.

Sumber daya perangkat

File di bawah `/dev`. Hanya perangkat karakter atau perangkat blok di bawah `/dev` diizinkan untuk sumber daya perangkat. Ini termasuk:

- Port serial yang digunakan untuk berkomunikasi dengan perangkat yang terhubung melalui port serial (sebagai contoh, `/dev/ttyS0`, `/dev/ttyS1`).
- USB yang digunakan untuk meghubungkan periferal USB (sebagai contoh, `/dev/ttyUSB0` atau `/dev/bus/usb`).
- GPIO digunakan untuk sensor dan aktuator melalui GPIO (sebagai contoh, `/dev/gpiomem`).
- GPU digunakan untuk mempercepat pembelajaran machine learning menggunakan GPU on-board (sebagai contoh, `/dev/nvidia0`).
- Kamera yang digunakan untuk menangkap gambar dan video (sebagai contoh, `/dev/video0`).

> **Note**
>
> `/dev/shm` adalah pengecualian. Hal ini dapat dikonfigurasi sebagai sumber daya volume saja. Sumber daya di bawah `/dev/shm` harus diberikan `rw` izin.

AWS IoT Greengrass juga mendukung jenis sumber daya yang digunakan untuk melakukan inference machine learning. Untuk informasi selengkapnya, lihat Lakukan inferensi machine learning.

Persyaratan

Persyaratan berikut berlaku untuk mengonfigurasi akses aman ke sumber daya lokal:

- Anda harus menggunakan AWS IoT Greengrass Core Software v1.3 atau yang lebih baru. Untuk membuat sumber daya untuk host direktori/proc, Anda harus menggunakan v1.6 atau yang lebih baru.
- Sumber daya lokal (termasuk driver yang diperlukan dan perpustakaan) harus diinstal dengan benar pada perangkat core Greengrass dan secara konsisten tersedia selama penggunaan.
- Operasi yang diinginkan dari sumber daya, dan akses ke sumber daya, tidak harus memerlukan hak istimewa root.
- Hanya `read` atau `read and write` izin tersedia. Fungsi Lambda tidak dapat melakukan operasi istimewa pada sumber daya.
- Anda harus menyediakan jalur lengkap sumber daya lokal pada sistem operasi perangkat core Greengrass.
- Nama sumber daya atau ID memiliki panjang maksimal 128 karakter dan harus menggunakan pola `[a-zA-Z0-9:_-]+`.

Sumber daya volume di bawah direktori /proc

Pertimbangan berikut berlaku untuk sumber daya volume yang berada di bawah host direktori /proc.

- Anda harus menggunakan AWS IoT Greengrass Core Software v1.6 atau yang lebih baru.
- Anda dapat mengizinkan akses hanya-baca untuk fungsi Lambda, tetapi tidak akses baca-tulis. Tingkat akses ini dikelola oleh AWS IoT Greengrass.
- Anda mungkin juga harus memberikan izin grup OS untuk mengaktifkan akses baca di sistem file. Sebagi contoh, andaikan direktori sumber atau file Anda memiliki 660 file izin, yang berarti bahwa hanya pemilik atau pengguna di kgrup telah membaca (dan menulis) akses. Dalam hal ini, Anda harus menambahkan izin pemilik grup OS ke sumber daya. Untuk informasi selengkapnya, lihat the section called “Izin akses file pemilik grup”.
- Lingkungan host dan namespace Lambda keduanya berisi direktori /proc, jadi pastikan untuk menghindari penamaan konflik ketika Anda menentukan jalur tujuan. Misalnya, jika /proc adalah

jalur sumber, Anda dapat menentukan /host-proc sebagai jalur tujuan (atau nama jalur selain "/proc").

Izin akses file pemilik grup

Proses AWS IoT Greengrass fungsi Lambda biasanya berjalan sebagai `ggc_user` dan `ggc_group`. Namun, Anda dapat memberikan izin akses file tambahan untuk proses fungsi Lambda dalam definisi sumber daya lokal, sebagai berikut:

- Untuk menambahkan izin dari grup Linux yang memiliki sumber daya, gunakan opsi konsol sumber`GroupOwnerSetting#AutoAddGroupOwner` daya atau Secara otomatis tambahkan izin sistem file dari grup sistem yang memiliki opsi konsol sumber daya.
- Untuk menambahkan izin dari grup Linux yang berbeda, gunakan`GroupOwnerSetting#GroupOwner` parameter atau Tentukan grup sistem lain untuk menambahkan izin dari grup Linux yang berbeda, gunakan opsi konsol izin sistem. Nilai `GroupOwner` diabaikan jika `GroupOwnerSetting#AutoAddGroupOwner` adalah benar.

Proses AWS IoT Greengrass fungsi Lambda mewarisi semua izin sistem file dari `ggc_user`, `ggc_group`, dan grup Linux (jika ditambahkan). Untuk fungsi Lambda untuk mengakses sumber daya, proses fungsi Lambda harus memiliki izin yang diperlukan untuk sumber daya. Anda dapat menggunakan `chmod(1)` perintah untuk mengubah izin dari sumber daya, jika perlu.

Lihat juga

- Service Quotas untuk sumber daya di Referensi Umum Amazon Web Services

Cara mengonfigurasi akses sumber daya lokal menggunakan AWS antarmuka baris perintah

Fitur ini tersedia untuk AWS IoT Greengrass Core v1.3 dan yang lebih baru.

Untuk menggunakan sumber daya lokal, Anda harus menambahkan definisi sumber daya untuk definisi grup yang di-deploy untuk perangkat core Greengrass Anda. Definisi grup juga harus berisi definisi fungsi Lambda di mana Anda memberikan izin akses untuk sumber daya lokal untuk fungsi

Lambda Anda. Untuk informasi lebih lanjut, termasuk persyaratan dan kendala, lihat Akses sumber daya lokal dengan fungsi dan konektor Lambda.

Tutorial ini menjelaskan proses untuk membuat sumber daya lokal dan mengonfigurasi akses ke sumber daya lokal menggunakan AWS Command Line Interface (CLI). Untuk mengikuti langkah-langkah dalam tutorial, Anda harus telah membuat grup Greengrass seperti yang dijelaskan di Mulai menggunakan AWS IoT Greengrass.

Untuk tutorial yang menggunakan AWS Management Console, lihat Cara mengonfigurasi akses sumber daya lokal menggunakan AWS Management Console.

Buat sumber daya lokal

Pertama, Anda menggunakan `CreateResourceDefinition` perintah untuk membuat definisi sumber daya yang menentukan sumber daya untuk diakses. Dalam contoh ini, kami membuat dua sumber daya, `TestDirectory` dan `TestCamera`:

```
aws greengrass create-resource-definition  --cli-input-json '{
    "Name": "MyLocalVolumeResource",
    "InitialVersion": {
        "Resources": [
            {
                "Id": "data-volume",
                "Name": "TestDirectory",
                "ResourceDataContainer": {
                    "LocalVolumeResourceData": {
                        "SourcePath": "/src/LRAtest",
                        "DestinationPath": "/dest/LRAtest",
                        "GroupOwnerSetting": {
                            "AutoAddGroupOwner": true,
                            "GroupOwner": ""
                        }
                    }
                }
            },
            {
                "Id": "data-device",
                "Name": "TestCamera",
                "ResourceDataContainer": {
                    "LocalDeviceResourceData": {
                        "SourcePath": "/dev/video0",
```

```
                        "GroupOwnerSetting": {
                            "AutoAddGroupOwner": true,
                            "GroupOwner": ""
                        }
                    }
                }
            }
        ]
    }
}'
```

Sumber Daya: Daftar `Resource` objek di grup Greengrass. Satu grup Greengrass dapat memiliki hingga 50 sumber daya.

Sumber Daya #Id: Pengenal unik sumber daya. ID digunakan untuk merujuk ke sumber daya dalam konfigurasi fungsi Lambda. Panjang maksimal 128 karakter. Pola: [a-zA-Z0-9:_-] +.

Sumber Daya #Name: Nama sumber daya. Nama sumber daya ditampilkan di konsol Greengrass. Panjang maksimal 128 karakter. Pola: [a-zA-Z0-9:_-] +.

LocalDeviceResourceData# SourcePath: Jalur absolut lokal dari sumber daya perangkat. Jalur sumber untuk sumber daya perangkat hanya dapat merujuk ke perangkat karakter atau perangkat blok di bawah `/dev`.

LocalVolumeResourceData# SourcePath: Jalur absolut lokal dari sumber daya volume pada perangkat inti Greengrass. Lokasi ini berada di luar kontainer yakni fungsi berjalan di. Path sumber untuk jenis sumber daya volume tidak dapat memulai dengan `/sys`.

LocalVolumeResourceData# DestinationPath: Jalur absolut sumber daya volume di dalam lingkungan Lambda. Lokasi ini adalah di dalam kontainer di mana fungsi berjalan.

GroupOwnerSetting: Memungkinkan Anda mengonfigurasi hak istimewa grup tambahan untuk proses Lambda. Bidang ini bersifat opsional. Untuk informasi selengkapnya, lihat Izin akses file pemilik grup.

GroupOwnerSetting# AutoAddGroupOwner: Jika benar, Greengrass secara otomatis menambahkan pemilik grup Linux OS yang ditentukan dari sumber daya ke hak istimewa proses Lambda. Dengan demikian proses Lambda memiliki izin akses file dari grup Linux yang ditambahkan.

GroupOwnerSetting# GroupOwner: Menentukan nama grup OS Linux yang hak istimewanya ditambahkan ke proses Lambda. Bidang ini bersifat opsional.

Sebuah versi definisi sumber daya ARN dikembalikan oleh `CreateResourceDefinition`. ARN harus digunakan ketika memperbarui definisi kelompok.

```
{
    "LatestVersionArn": "arn:aws:greengrass:us-west-2:012345678901:/greengrass/
definition/resources/ab14d0b5-116e-4951-a322-9cde24a30373/versions/a4d9b882-
d025-4760-9cfe-9d4fada5390d",
    "Name": "MyLocalVolumeResource",
    "LastUpdatedTimestamp": "2017-11-15T01:18:42.153Z",
    "LatestVersion": "a4d9b882-d025-4760-9cfe-9d4fada5390d",
    "CreationTimestamp": "2017-11-15T01:18:42.153Z",
    "Id": "ab14d0b5-116e-4951-a322-9cde24a30373",
    "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/resources/
ab14d0b5-116e-4951-a322-9cde24a30373"
}
```

Buat fungsi Greengrass

Setelah sumber daya dibuat, gunakan `CreateFunctionDefinition` perintah untuk membuat fungsi Greengrass dan berikan fungsi akses ke sumber daya:

```
aws greengrass create-function-definition --cli-input-json '{
    "Name": "MyFunctionDefinition",
    "InitialVersion": {
        "Functions": [
            {
                "Id": "greengrassLraTest",
                "FunctionArn": "arn:aws:lambda:us-
west-2:012345678901:function:lraTest:1",
                "FunctionConfiguration": {
                    "Pinned": false,
                    "MemorySize": 16384,
                    "Timeout": 30,
                    "Environment": {
                        "ResourceAccessPolicies": [
                            {
                                "ResourceId": "data-volume",
                                "Permission": "rw"
                            },
                            {
                                "ResourceId": "data-device",
                                "Permission": "ro"
```

```
                        }
                    ],
                    "AccessSysfs": true
                }
            }
        }
    ]
}
}'
```

ResourceAccessPolicies: Berisi `resourceId` dan `permission` yang memberikan akses fungsi Lambda ke sumber daya. Sebuah fungsi Lambda dapat mengakses maksimal 20 sumber daya.

ResourceAccessPolicy#Permission: Menentukan izin yang dimiliki fungsi Lambda pada sumber daya. Pilihan yang tersedia adalah `rw` (baca/tulis) atau `ro` (hanya baca).

AccessSysfs: Jika benar, proses Lambda dapat memiliki akses baca ke `/sys` folder pada perangkat inti Greengrass. Ini digunakan dalam kasus-kasus di mana fungsi Lambda Greengrass perlu membaca informasi perangkat dari `/sys`.

Sekali lagi, `CreateFunctionDefinition` kembalikan fungsi definisi versi ARN. ARN harus digunakan dalam versi definisi grup Anda.

```
{
    "LatestVersionArn": "arn:aws:greengrass:us-west-2:012345678901:/greengrass/
definition/functions/3c9b1685-634f-4592-8dfd-7ae1183c28ad/versions/37f0d50e-ef50-4faf-
b125-ade8ed12336e",
    "Name": "MyFunctionDefinition",
    "LastUpdatedTimestamp": "2017-11-22T02:28:02.325Z",
    "LatestVersion": "37f0d50e-ef50-4faf-b125-ade8ed12336e",
    "CreationTimestamp": "2017-11-22T02:28:02.325Z",
    "Id": "3c9b1685-634f-4592-8dfd-7ae1183c28ad",
    "Arn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/
functions/3c9b1685-634f-4592-8dfd-7ae1183c28ad"
}
```

Tambahkan fungsi Lambda ke grup

Akhirnya, gunakan `CreateGroupVersion` untuk menambahkan fungsi ke grup. Sebagai contoh:

```
aws greengrass create-group-version --group-id "b36a3aeb-3243-47ff-9fa4-7e8d98cd3cf5" \
```

```
--resource-definition-version-arn "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/resources/db6bf40b-29d3-4c4e-9574-21ab7d74316c/versions/31d0010f-
e19a-4c4c-8098-68b79906fb87" \
--core-definition-version-arn "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/cores/adbf3475-f6f3-48e1-84d6-502f02729067/
versions/297c419a-9deb-46dd-8ccc-341fc670138b" \
--function-definition-version-arn "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/functions/d1123830-da38-4c4c-a4b7-e92eec7b6d3e/versions/a2e90400-
caae-4ffd-b23a-db1892a33c78" \
--subscription-definition-version-arn "arn:aws:greengrass:us-west-2:123456789012:/
greengrass/definition/subscriptions/7a8ef3d8-1de3-426c-9554-5b55a32fbcb6/
versions/470c858c-7eb3-4abd-9d48-230236bfbf6a"
```

> **Note**
>
> Untuk mempelajari bagaimana mendapatkan ID grup untuk digunakan dengan perintah ini, lihat the section called “Mendapatkan ID grup”.

Sebuah versi grup baru dikembalikan:

```
{
    "Arn": "arn:aws:greengrass:us-west-2:012345678901:/greengrass/groups/
b36a3aeb-3243-47ff-9fa4-7e8d98cd3cf5/versions/291917fb-ec54-4895-823e-27b52da25481",
    "Version": "291917fb-ec54-4895-823e-27b52da25481",
    "CreationTimestamp": "2017-11-22T01:47:22.487Z",
    "Id": "b36a3aeb-3243-47ff-9fa4-7e8d98cd3cf5"
}
```

Grup Greengrass Anda sekarang berisi fungsi Lambda LraTest yang memiliki akses ke dua sumber daya: dan. TestDirectory TestCamera

Contoh ini fungsi Lambda, `lraTest.py`, ditulis dengan Python, menulis ke sumber daya volume lokal:

```
# Demonstrates a simple use case of local resource access.
# This Lambda function writes a file test to a volume mounted inside
# the Lambda environment under destLRAtest. Then it reads the file and
# publishes the content to the AWS IoT LRAtest topic.

import sys
```

```
import greengrasssdk
import platform
import os
import logging

# Setup logging to stdout
logger = logging.getLogger(__name__)
logging.basicConfig(stream=sys.stdout, level=logging.DEBUG)

# Create a Greengrass Core SDK client.
client = greengrasssdk.client('iot-data')
volumePath = '/dest/LRAtest'

def function_handler(event, context):
    try:
        client.publish(topic='LRA/test', payload='Sent from AWS IoT Greengrass Core.')
        volumeInfo = os.stat(volumePath)
        client.publish(topic='LRA/test', payload=str(volumeInfo))
        with open(volumePath + '/test', 'a') as output:
            output.write('Successfully write to a file.')
        with open(volumePath + '/test', 'r') as myfile:
            data = myfile.read()
        client.publish(topic='LRA/test', payload=data)
    except Exception as e:
        logger.error('Failed to publish message: ' + repr(e))
    return
```

Perintah ini disediakan oleh API Greengrass untuk membuat dan mengelola definisi sumber daya dan versi definisi sumber daya:

- CreateResourceDefinition
- CreateResourceDefinitionVersion
- DeleteResourceDefinition
- GetResourceDefinition
- GetResourceDefinitionVersion
- ListResourceDefinitions
- ListResourceDefinitionVersions
- UpdateResourceDefinition

Memecahkan masalah

- Q: Mengapa deployment grup Greengrass saya gagal dengan kesalahan yang mirip dengan:

  ```
  group config is invalid:
      ggc_user or [ggc_group root tty] don't have ro permission on the file: /dev/tty0
  ```

  A: Kesalahan ini menunjukkan bahwa proses Lambda tidak memiliki izin untuk mengakses sumber daya tertentu. Solusinya adalah mengubah izin file dari sumber daya sehingga Lambda dapat mengaksesnya. (Lihat Izin akses file pemilik grup untuk detailnya).

- Q: Ketika saya mengonfigurasi `/var/run` sebagai sumber volume, mengapa fungsi Lambda gagal untuk memulai dengan pesan salah di runtime.log:

  ```
  [ERROR]-container_process.go:39,Runtime execution error: unable to start lambda
   container.
  container_linux.go:259: starting container process caused "process_linux.go:345:
  container init caused \"rootfs_linux.go:62: mounting \\\"/var/run\\\" to rootfs \\\"/
  greengrass/ggc/packages/1.3.0/rootfs_sys\\\" at \\\"/greengrass/ggc/packages/1.3.0/
  rootfs_sys/run\\\"
  caused \\\"invalid argument\\\"\""
  ```

  A: AWS IoT Greengrass core saat ini tidak mendukung konfigurasi `/var`, `/var/run`, dan `/var/lib` sebagai sumber daya volume. Satu solusi adalah untuk pertama me-mount `/var`, `/var/run` atau `/var/lib` di folder yang berbeda dan kemudian mengonfigurasi folder sebagai sumber volume.

- Q: Ketika saya mengonfigurasi `/dev/shm` sebagai sumber volume dengan izin baca-saja, mengapa fungsi Lambda gagal untuk memulai dengan kesalahan dalam runtime.log:

  ```
  [ERROR]-container_process.go:39,Runtime execution error: unable to start lambda
   container.
  container_linux.go:259: starting container process caused "process_linux.go:345:
  container init caused \"rootfs_linux.go:62: mounting \\\"/dev/shm\\\" to rootfs \\\"/
  greengrass/ggc/packages/1.3.0/rootfs_sys\\\" at \\\"/greengrass/ggc/packages/1.3.0/
  rootfs_sys/dev/shm\\\"
  caused \\\"operation not permitted\\\"\""”
  ```

  A: `/dev/shm` hanya dapat dikonfigurasi sebagai baca/tulis. Ubah izin sumber daya untuk `rw` untuk menyelesaikan masalah.

Cara mengonfigurasi akses sumber daya lokal menggunakan AWS Management Console

Fitur ini tersedia untuk AWS IoT Greengrass Core v1.3 dan yang lebih baru.

Anda dapat mengonfigurasi fungsi Lambda agar aman mengakses sumber daya lokal pada host perangkat core Greengrass. Sumber daya lokal merujuk ke bus dan periferal yang ada secara fisik pada host, atau volume sistem file pada OS host. Untuk informasi lebih lanjut, termasuk persyaratan dan kendala, lihat Akses sumber daya lokal dengan fungsi dan konektor Lambda.

Tutorial ini menjelaskan cara menggunakan AWS Management Console untuk mengonfigurasi akses ke sumber daya lokal yang ada di perangkat AWS IoT Greengrass core. Itu berisi langkah-langkah tingkat tinggi berikut:

1. Buat paket deployment fungsi Lambda
2. Buat dan publikasi fungsi Lambda
3. Menambahkan fungsi Lambda ke grup
4. Menambahkan sumber daya lokal ke grup
5. Menambahkan langganan ke grup
6. Men-deploy grup

Untuk tutorial yang menggunakan AWS Command Line Interface, lihat Cara mengonfigurasi akses sumber daya lokal menggunakan AWS antarmuka baris perintah.

Prasyarat

Untuk menyelesaikan tutorial ini, Anda memerlukan:

- Sebuah grup Greengrass dan core Greengrass (v1.3 atau lebih baru). Untuk membuat grup Greengrass atau core, lihat Mulai menggunakan AWS IoT Greengrass.
- Direktori berikut ada pada perangkat core Greengrass:
  - /src/LRAtest
  - /dest/LRAtest

  Grup pemilik direktori ini harus memiliki akses baca dan tulis ke direktori. Anda dapat menggunakan perintah berikut untuk memberikan akses:

```
sudo chmod 0775 /src/LRAtest
```

Langkah 1: Buat paket deployment fungsi Lambda

Pada langkah ini, Anda membuat paket deployment fungsi Lambda, yang merupakan file ZIP yang berisi kode fungsi dan dependensi. Anda juga mengunduh AWS IoT Greengrass Core SDK untuk memasukkan dalam paket sebagai dependensi.

1. Pada komputer Anda, salin skrip Python berikut ke file lokal bernama `lraTest.py`. Ini adalah logika aplikasi untuk fungsi Lambda.

```
# Demonstrates a simple use case of local resource access.
# This Lambda function writes a file test to a volume mounted inside
# the Lambda environment under destLRAtest. Then it reads the file and
# publishes the content to the AWS IoT LRAtest topic.

import sys
import greengrasssdk
import platform
import os
import logging

# Setup logging to stdout
logger = logging.getLogger(__name__)
logging.basicConfig(stream=sys.stdout, level=logging.DEBUG)

# Create a Greengrass Core SDK client.
client = greengrasssdk.client('iot-data')
volumePath = '/dest/LRAtest'

def function_handler(event, context):
    try:
        client.publish(topic='LRA/test', payload='Sent from AWS IoT Greengrass
 Core.')
        volumeInfo = os.stat(volumePath)
        client.publish(topic='LRA/test', payload=str(volumeInfo))
        with open(volumePath + '/test', 'a') as output:
            output.write('Successfully write to a file.')
        with open(volumePath + '/test', 'r') as myfile:
            data = myfile.read()
```

```
        client.publish(topic='LRA/test', payload=data)
    except Exception as e:
        logger.error('Failed to publish message: ' + repr(e))
    return
```

2. Dari halaman AWS IoT Greengrass Core SDK mengunduh, unduh AWS IoT Greengrass Core SDK for Python ke komputer Anda.
3. Unzip paket yang diunduh untuk mendapatkan SDK. SDK adalah `greengrasssdk` folder.
4. Zip item berikut ke dalam file bernama `lraTestLambda.zip`:

   - `lraTest.py`. Logika aplikasi.
   - `greengrasssdk`. Diperlukan perpustakaan untuk semua fungsi Python Lambda.

   File `lraTestLambda.zip` adalah paket deployment fungsi Lambda Anda. Sekarang Anda siap untuk membuat fungsi Lambda dan mengunggah paket deployment.

Langkah 2: Buat dan publikasikan fungsi Lambda

Pada langkah ini, Anda menggunakan konsol AWS Lambda untuk membuat fungsi Lambda dan mengonfigurasinya agar dapat menggunakan paket deployment Anda. Kemudian, Anda mempublikasikan versi fungsi dan membuat alias.

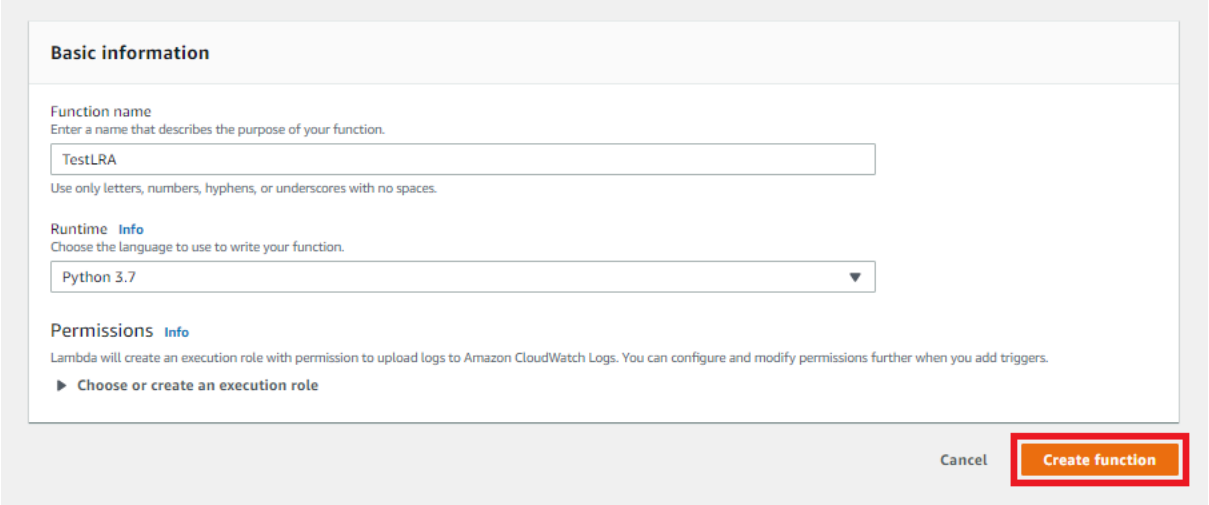
Pertama, buat fungsi Lambda.

1. Di AWS Management Console, pilih Layanan, dan buka konsol AWS Lambda tersebut.
2. PilihFungsi.
3. Pilih Buat fungsi dan kemudian pilih Tulis dari awal.
4. Di bagian Informasi dasar tersebut, gunakan nilai-nilai berikut.

   a. Untuk Nama fungsi, masukkan **`TestLRA`**.

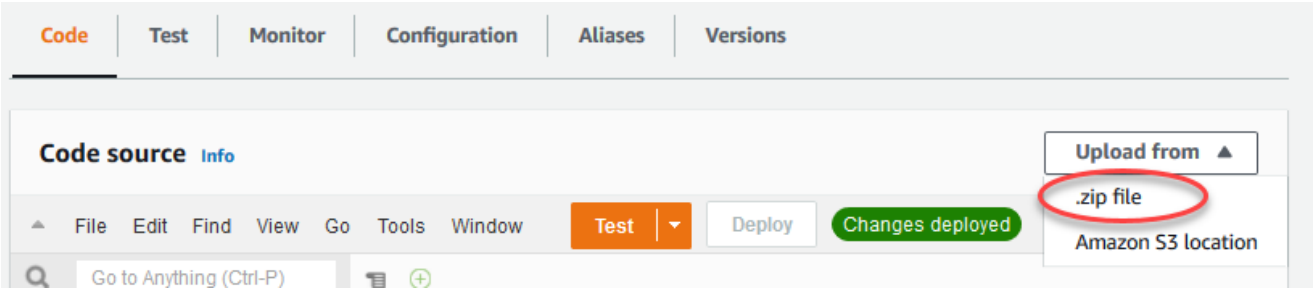
   b. Untuk Waktu pengoperasian, pilih Python 3.7.

   c. Untuk Izin, pertahankan pengaturan default. Hal ini menciptakan peran eksekusi yang memberikan izin Lambda basic. Peran ini tidak digunakan oleh AWS IoT Greengrass.
5. Pilih Buat fungsi.



6. Unggah paket deployment fungsi Lambda Anda dan daftarkan handler-nya.

   a. Pada tab Kode tersebut, di bawah Sumber kode, pilih Unggah dari. Dari dropdown, pilih file .zip.



   b. Pilih Unggah, lalu pilih paket deployment `lraTestLambda.zip` Anda. Lalu, pilih Simpan.

   c. Pada tab Kode fungsi, di bawah Pengaturan waktu aktif, pilih Edit, dan kemudian masukkan nilai-nilai berikut.

      - Untuk Waktu aktif, pilih Python 3.7.
      - Untuk Handler, memasukkan lraTest.function_Handler.

   d. Pilih Save (Simpan).

> **Note**
>
> Tombol Tes pada konsol AWS Lambda tidak bekerja dengan fungsi ini. Pada AWS IoT Greengrass Core SDK tidak berisi modul yang diperlukan untuk menjalankan fungsi Greengrass Lambda Anda secara independen di konsol AWS Lambda

tersebut. Modul-modul ini (misalnya, `greengrass_common`) dipasok ke fungsi setelah mereka di-deploy ke core Greengrass Anda.

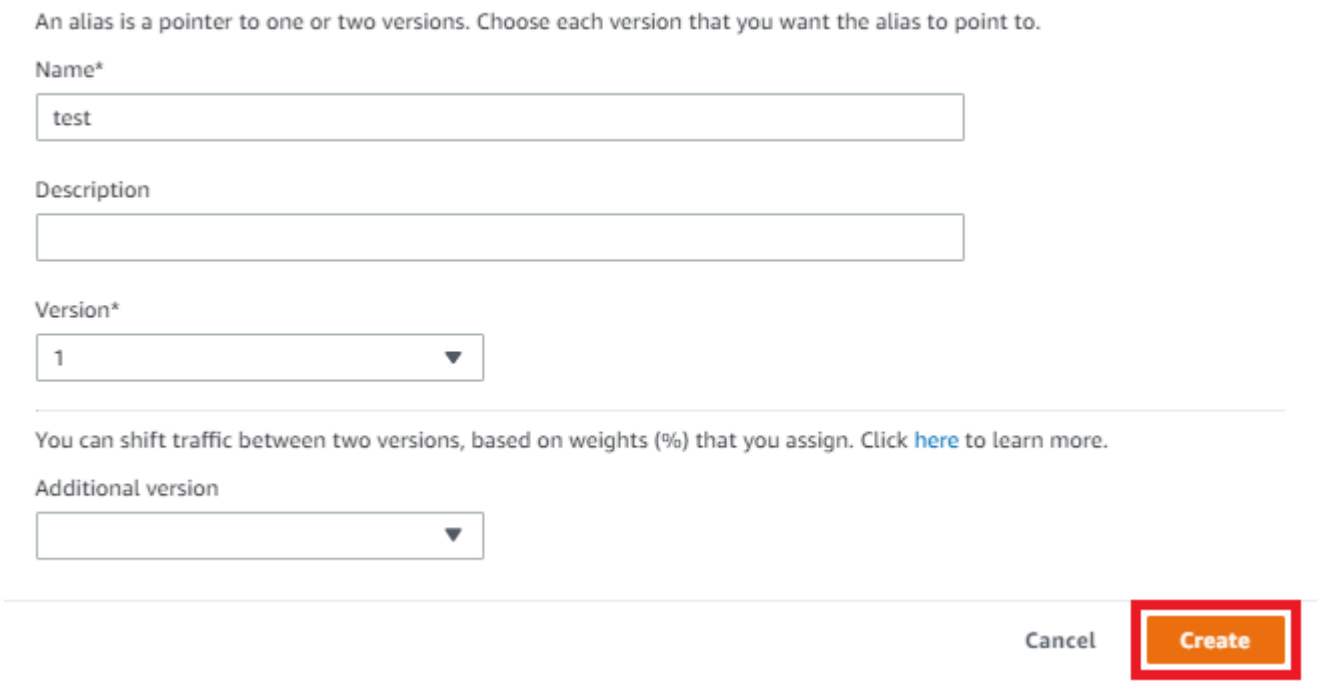
Selanjutnya, mempublikasikan versi pertama fungsi Lambda Anda. Kemudian, buat alias untuk versi.

Grup Greengrass dapat mereferensi fungsi Lambda dengan alias (direkomendasikan) atau dengan versi. Menggunakan alias membuatnya lebih mudah untuk mengelola pembaruan kode karena Anda tidak perlu mengubah tabel langganan atau definisi grup ketika kode fungsi diperbarui. Sebaliknya, Anda hanya mengarahkan alias ke versi fungsi baru.

7. Dari Tindakan, pilih Terbitkan versi baru.
8. Untuk Deskripsi versi, memasukkan **`First version`**, lalu pilih Publikasikan.
9. Pada halama konfigurasi TestLRA: 1 tersebut, dari Tindakan, pilih Buat alias.
10. PadaMembuat aliashalaman, untukNamaENTER**`test`**. Untuk Versi, masukkan 1.

> **Note**
>
> AWS IoT Greengrass tidak mendukung alias Lambda dari versi $TERBARU ini.

11. Pilih Create (Buat).



Anda sekarang dapat menambahkan fungsi Lambda ke grup Greengrass Anda.

Langkah 3: Tambahkan fungsi Lambda ke grup Greengrass

Pada langkah ini, Anda menambahkan fungsi ke grup Anda dan mengonfigurasi siklus hidup fungsi.

Pertama, tambahkan fungsi Lambda ke grup Greengrass Anda.

1. DiAWS IoTpanel navigasi konsol, di bawahKelolaPerluasPerangkat Greengrass, dan kemudian pilihGrup (V1).
2. Pilih grup Greengrass di mana Anda ingin menambahkan fungsi Lambda.
3. Pada halaman Konfigurasi grup, pilihFungsi LambdaTab.
4. Di bawahFungsi Lambda sayabagian, pilihTambahkan.
5. PadaTambahkan fungsi Lambdahalaman, pilihFungsi Lambda. Pilih **`TestLRA`**.
6. PilihVersi fungsi Lambda.
7. DiKonfigurasi fungsi Lambdabagian, pilihPengguna dan grup sistemdanKontainerisasi fungsi Lambda.

   Selanjutnya, konfigurasikan siklus hidup fungsi Lambda.
8. UntukWaktu habis, choose30 detik.

   > **Important**
   >
   > Fungsi Lambda yang menggunakan sumber daya lokal (seperti yang dijelaskan dalam prosedur ini) harus berjalan dalam kontainer Greengrass. Jika tidak, deployment gagal jika Anda mencoba untuk men-deploy fungsi. Untuk informasi lebih lanjut, lihat Kontainerisasi.
9. Di bagian bawah halaman, pilihTambahkan fungsi Lambda.

Langkah 4: Tambahkan sumber daya lokal ke grup Greengrass

Pada langkah ini, Anda menambahkan sumber daya volume lokal untuk grup Greengrass dan memberikan fungsi akses membaca dan menulis ke sumber daya. Sumber daya lokal memiliki

lingkup grup-tingkat. Anda dapat memberikan izin untuk setiap fungsi Lambda dalam grup untuk mengakses sumber daya.

1. Pada halaman Konfigurasi grup, pilihSumber dayaTab.
2. Di bawahSumber daya lokalbagian, pilihTambahkan.
3. PadaTambahkan sumber daya lokalhalaman, gunakan nilai-nilai berikut.

   a. Untuk Nama sumber daya, memasukkan **`testDirectory`**.
   b. Untuk Jenis sumber daya, pilih Volume.
   c. UntukJalur perangkatENTER**`/src/LRAtest`**. Jalur ini harus ada pada OS host.

      Jalur perangkat lokal adalah jalur absolut lokal sumber daya pada sistem file dari perangkat core. Lokasi ini berada di luar Kontainer di mana fungsi berjalan. Jalur tidak dapat dimulai dengan `/sys`.
   d. Untuk Jalur tujuan, masukkan **`/dest/LRAtest`**. Jalur ini harus ada pada OS host.

      Jalur tujuan adalah jalur absolut dari sumber daya di namespace Lambda. Lokasi ini adalah di dalam kontainer di mana fungsi berjalan.
   e. Di bawahPemilik grup sistem dan izin akses filechooseSecara otomatis menambahkan izin sistem file dari grup sistem yang memiliki sumber daya.

      YangPemilik grup sistem dan izin akses filePilihan memungkinkan Anda memberikan izin akses file tambahan untuk proses Lambda. Untuk informasi selengkapnya, lihat Izin akses file pemilik grup.
4. PilihTambahkan sumber daya. Halaman Sumber Daya menampilkan sumber daya testDirectory baru.

Langkah 5: Tambahkan langganan ke grup Greengrass

Pada langkah ini, Anda akan menambahkan dua langganan ke grup Greengrass. Langganan ini memungkinkan komunikasi dua arah antara fungsi Lambda dan AWS IoT.

Pertama, buat langganan untuk fungsi Lambda agar mengirim pesan ke AWS IoT.

1. Pada halaman Konfigurasi grup, pilihLangganantab.
2. Pilih Tambahkan.
3. PadaBuat langgananhalaman, konfigurasikan sumber dan target, sebagai berikut:

a. UntukJenis Sumber, chooseFungsi Lambda, dan kemudian pilihTestLRA.

b. UntukJenis target, chooseLayanan, dan kemudian pilihIoT Cloud.

c. UntukFilter topikENTER**`LRA/test`**, dan kemudian pilihBuat langganan.

4. Halaman Berlangganan menampilkan langganan baru.

Selanjutnya, konfigurasikan langganan yang memanggil fungsi dari AWS IoT.

5. Di halaman Langganan tersebut, pilih Tambahkan langganan.
6. Pada halaman Pilih sumber dan target tersebut, mengonfigurasi sumber dan target, sebagai berikut:

a. UntukJenis Sumber, chooseFungsi Lambda, dan kemudian pilihIoT Cloud.

b. UntukJenis target, chooseLayanan, dan kemudian pilihTestLRA.

c. Pilih Selanjutnya.

7. Pada halaman Filter data Anda dengan topik ini, di Filter topik, masukkan **`invoke/LRAFunction`**, lalu pilih Selanjutnya.
8. Pilih Selesai. Halaman Berlangganan menampilkan kedua langganan.

Langkah 6: DeployAWS IoT Greengrasskelompok

Pada langkah ini, Anda men-deploy versi definisi grup.

1. Pastikan bahwa AWS IoT Greengrass core sedang berjalan. Jalankan perintah berikut di terminal Raspberry Pi Anda, sesuai kebutuhan.

a. Untuk memeriksa apakah daemon sedang berjalan:

```
ps aux | grep -E 'greengrass.*daemon'
```

Jika outputnya berisi entri `root` untuk `/greengrass/ggc/packages/1.11.6/bin/daemon`, maka daemon sedang berjalan.

> **Note**
>
> Versi di jalur tergantung pada versi perangkat lunak AWS IoT Greengrass core yang diinstal pada perangkat core Anda.

b. Untuk memulai daemon:

```
cd /greengrass/ggc/core/
sudo ./greengrassd start
```

2. Pada halaman Konfigurasi grup, pilihDeploy.

> **Note**
>
> Deployment gagal jika Anda menjalankan fungsi Lambda Anda tanpa kontainerisasi dan mencoba untuk mengakses sumber daya lokal terlampir.

3. Jika diminta, padaFungsi Lambdatab, di bawahFungsi Lambda sistemchooseDetektor IP, dan kemudianedit, dan kemudianMendeteksi.

   Hal ini mengaktifkan perangkat untuk secara otomatis memperoleh informasi konektivitas untuk core, seperti alamat IP, DNS, dan nomor port. Deteksi otomatis direkomendasikan, namun AWS IoT Greengrass juga support titik akhir yang ditentukan secara manual. Anda hanya diminta untuk metode penemuan pertama kalinya bahwa grup di-deploy.

   > **Note**
   >
   > Jika diminta, berikan izin untuk membuat Peran layanan Greengrass dan kaitkan dengan Akun AWS Anda pada Wilayah AWS. Peran ini memungkinkan AWS IoT Greengrass untuk mengakses sumber daya Anda di layanan AWS ini.

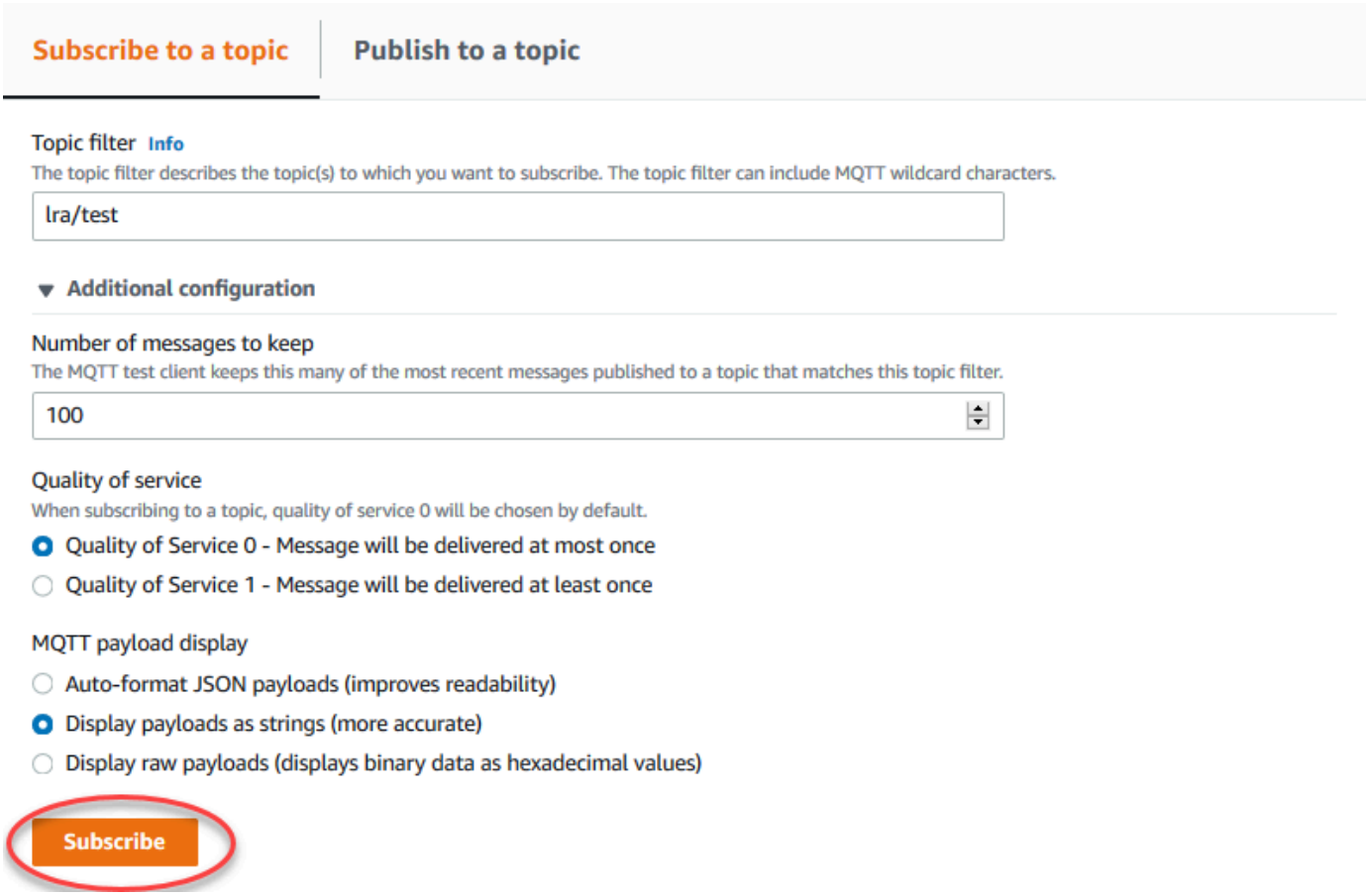
   Halaman Deployment menampilkan timestamp deployment, ID versi, dan status. Setelah selesai, status deploymentCompleted (Lengkap).

   Untuk bantuan penyelesaian masalah, lihat Memecahkan masalah.

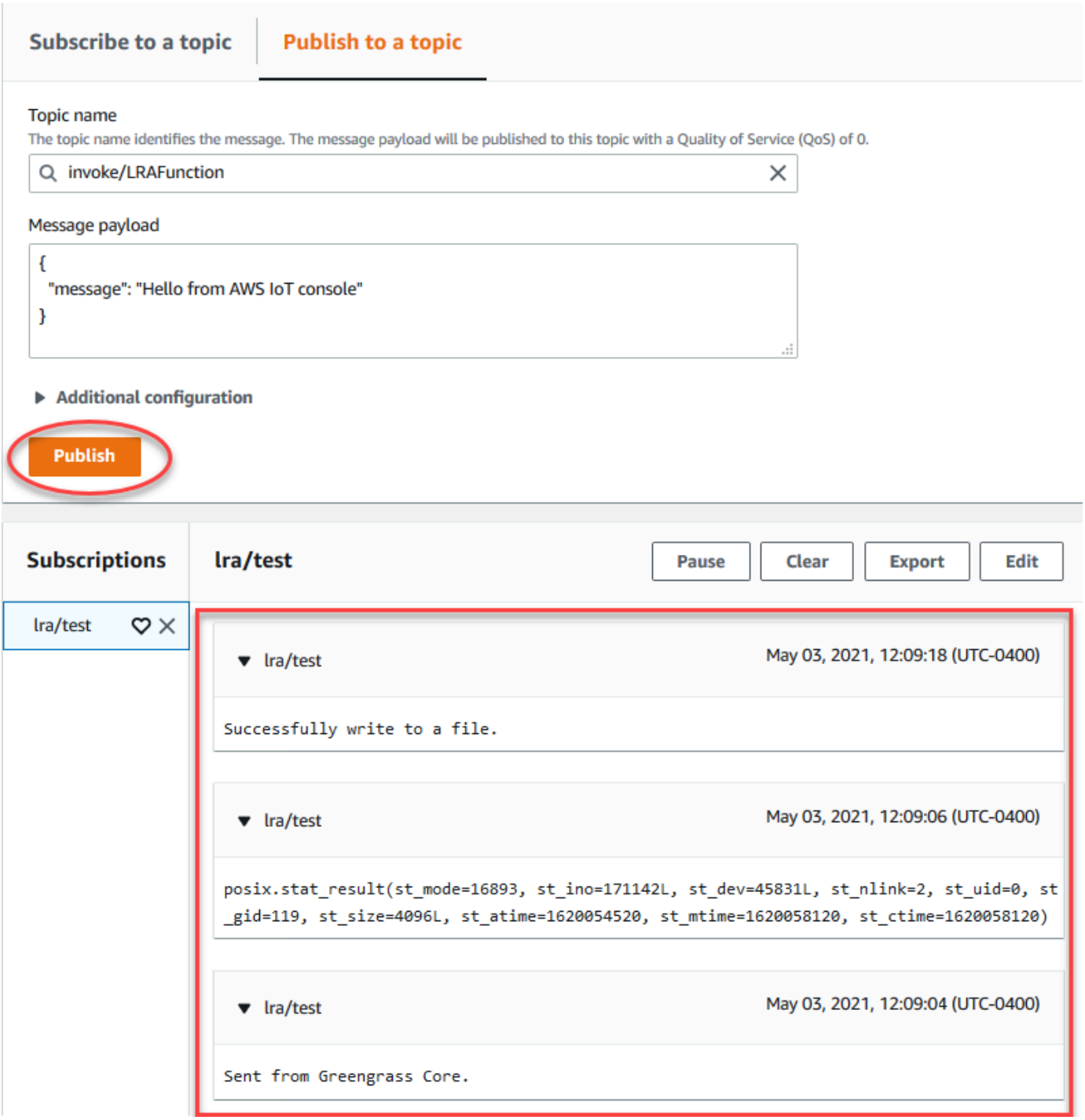
Uji akses sumber daya lokal

Sekarang Anda dapat memverifikasi apakah akses sumber daya lokal dikonfigurasi dengan benar. Untuk menguji, Anda berlangganan topik `LRA/test` dan mempublikasikan ke `invoke/LRAFunction` topik. Tes ini berhasil jika fungsi Lambda mengirimkan muatan yang diharapkan ke AWS IoT.

1. DariAWS IoTmenu navigasi konsol, di bawahPengujian, chooseKlien pengujian MQTT.
2. Di bawahBerlangganan topik, untukFilter topikENTER**`LRA/test`**.
3. Di bawahInformasi tambahan, untukTampilan muatan MQTTchooseTampilkan muatan sebagai string.
4. Pilih Langganan. Fungsi Lambda Anda menerbitkan untuk topik LRA/tes.



5. Di bawahMenerbitkan ke topik, dalamNama topikmemasukkan**`invoke/LRAFunction`**, dan kemudian pilihPublikasikanuntuk memanggil fungsi Lambda Anda. Tes ini berhasil jika halaman menampilkan muatan tiga pesan fungsi.



File tes yang dibuat oleh fungsi Lambda ada di direktori `/src/LRAtest` pada perangkat core Greengrass. Meskipun fungsi Lambda menulis ke file dalam direktori `/dest/LRAtest` tersebut, file tersebut terlihat hanya di namespace Lambda. Anda tidak dapat melihatnya di namespace Linux biasa. Setiap perubahan ke jalur tujuan tercermin dalam jalur sumber pada sistem file.

Untuk bantuan penyelesaian masalah, lihat Memecahkan masalah.

Lakukan inferensi machine learning

Fitur ini tersedia untuk Core AWS IoT Greengrass v1.6 atau yang lebih baru.

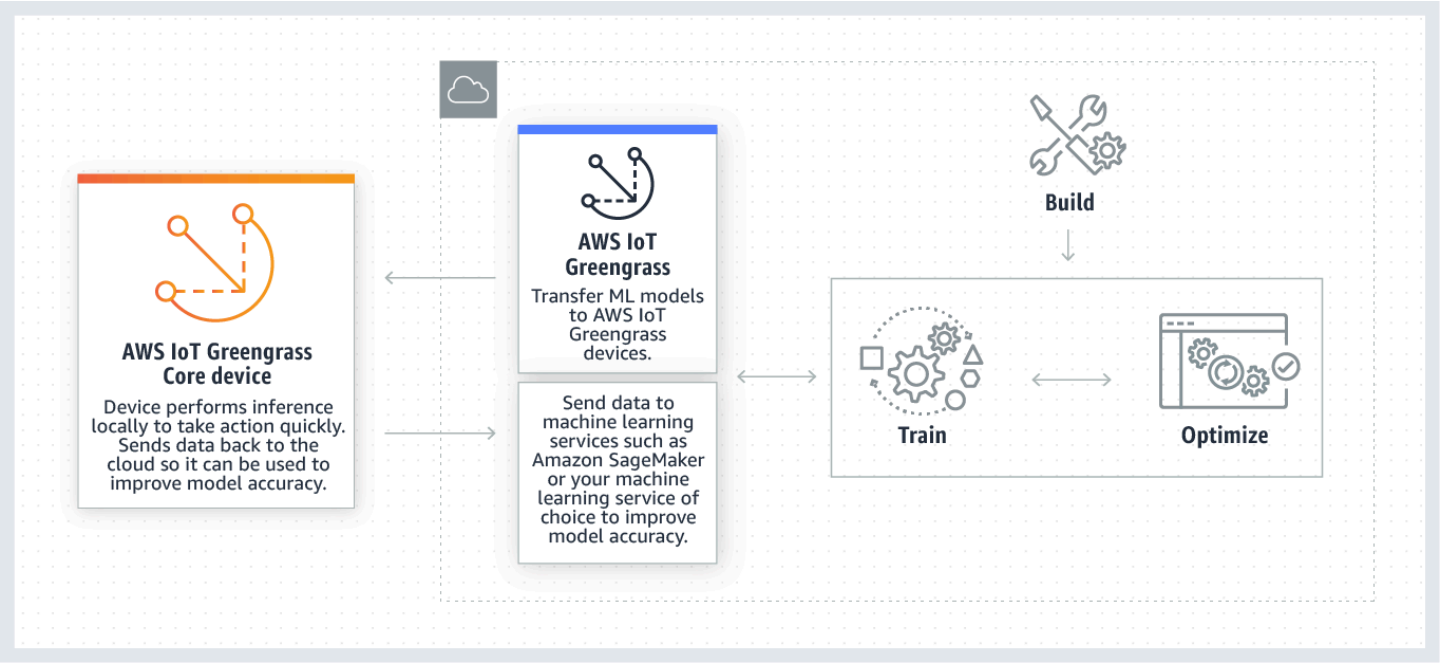
Dengan AWS IoT Greengrass, Anda dapat menampilkan inferensi machine learning (ML) di edge pada data yang dihasilkan secara lokal menggunakan model cloud-trained. Anda mendapatkan keuntungan dari latensi rendah dan penghematan biaya untuk menjalankan inferensi lokal, namun masih memanfaatkan daya komputasi cloud untuk model pelatihan dan pemrosesan yang rumit.

Untuk memulai melakukan inferensi lokal, lihat the section called “Cara mengonfigurasi inferensi machine learning”.

Bagaimana inferensi ML AWS IoT Greengrass bekerja

Anda dapat melatih model inferensi Anda di mana saja, men-deploy secara lokal sebagai sumber daya machine learning di grup Greengrass, dan kemudian mengaksesnya dari fungsi Greengrass Lambda. Sebagai contoh, Anda dapat membangun dan melatih model pembelajaran mendalam di SageMaker dan men-deploy mereka ke Greengrass core Anda. Kemudian, fungsi Lambda Anda dapat menggunakan model lokal untuk melakukan inferensi pada perangkat yang terhubung dan mengirim data pelatihan baru kembali ke cloud.

Diagram berikut menunjukkan AWS IoT Greengrass alur kerja kesimpulan ML.



AWS IoT Greengrass Inferensi ML menyederhanakan setiap langkah dari alur kerja ML, termasuk:

- Membangun dan deploying prototipe kerangka ML.
- Mengakses model cloud-trained dan deploying mereka ke perangkat Greengrass core.
- Membuat aplikasi inferensi yang dapat mengakses akselerator perangkat keras (seperti GPU dan FPGA) sebagai sumber daya lokal.

Sumber daya machine learning

Sumber daya machine learning mewakili model inferensi cloud-trained yang di-deploy ke core AWS IoT Greengrass ini. Untuk men-deploy sumber daya machine learning, pertama Anda tambahkan sumber daya ke grup Greengrass, dan kemudian Anda tentukan bagaimana fungsi Lambda dalam grup dapat mengaksesnya. Selama deployment grup, AWS IoT Greengrass ambil paket model sumber dari cloud dan ekstrak mereka ke direktori di dalam namespace waktu aktif Lambda. Kemudian, fungsi Greengrass Lambda menggunakan model lokal di-deploy untuk melakukan inferensi.

Untuk memperbarui model lokal yang di-deploy, pertama perbarui model sumber (di cloud) yang sesuai dengan sumber daya machine learning, dan kemudian men-deploy grup. Selama deployment, AWS IoT Greengrass memeriksa sumber untuk perubahan. Jika perubahan terdeteksi, maka AWS IoT Greengrass memperbarui model lokal.

Sumber model yang didukung

AWS IoT Greengrassmendukung SageMaker dan sumber model Amazon S3 untuk sumber daya machine learning.

Persyaratan berikut berlaku untuk sumber model:

- Ember S3 yang menyimpan SageMaker Sumber model Amazon S3 tidak harus dienkripsi menggunakan SSE-C. Untuk bucket yang menggunakan enkripsi sisi server,AWS IoT GreengrassInferensi saat ini mendukung opsi enkripsi SSE-S3 atau SSE-KMS saja. Untuk informasi selengkapnya tentang opsi enkripsi sisi server, lihatMelindungi data menggunakan enkripsi sisi serverdiPanduan Pengguna Amazon Simple Storage Service.
- Nama-nama ember S3 yang menyimpan SageMaker sumber model Amazon S3 tidak harus mencakup periode (`.`). Untuk informasi selengkapnya, lihat aturan tentang penggunaan bucket hosted-style dengan SSL diAturan penamaan bucketdiPanduan Pengguna Amazon Simple Storage Service.

- Tingkat layanan dukungan Wilayah AWS harus tersedia untuk AWS IoT Greengrass dan SageMaker. Saat ini,AWS IoT Greengrassmendukung SageMaker model di Wilayah berikut ini:
  - AS Timur (Ohio)
  - US East (N. Virginia)
  - US West (Oregon)
  - Asia Pacific (Mumbai)
  - Asia Pacific (Seoul)
  - Asia Pacific (Singapore)
  - Asia Pacific (Sydney)
  - Asia Pacific (Tokyo)
  - Europe (Frankfurt)
  - Europe (Ireland)
  - Europe (London)
- AWS IoT Greengrass harus memiliki izin `read` ke sumber model, seperti yang dijelaskan di bagian berikut.

SageMaker

AWS IoT Greengrassmendukung model yang disimpan sebagai SageMaker pekerjaan pelatihan. SageMaker adalah layanan ML-nya yang dikelola sepenuhnya yang dapat Anda gunakan untuk membangun dan melatih model menggunakan algoritma built-in atau kustom. Untuk informasi lebih lanjut, lihat Apakah SageMaker? dalam Panduan Developer SageMaker.

Jika Anda mengonfigurasi SageMaker lingkungan olehmembuat emberyang namanya berisi`sagemaker`, makaAWS IoT Greengrassmemiliki izin yang cukup untuk mengakses SageMaker pekerjaan pelatihan. Kebijakan terkelola `AWSGreengrassResourceAccessRolePolicy` mengizinkan akses ke bucket yang namanya berisi string `sagemaker`. Kebijakan ini terlampir pada peran layanan Greengrass.

Jika tidak, Anda harus memberikan izin AWS IoT Greengrass `read` ke bucket tempat tugas pelatihan Anda disimpan. Untuk melakukannya, menanamkan kebijakan inline berikut di peran layanan. Anda dapat daftar beberapa bucket ARN.

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket-name"
            ]
        }
    ]
}
```

Amazon S3

AWS IoT Greengrass mendukung model yang disimpan di Amazon S3 sebagai `tar.gz` atau `.zip` file.

Untuk mengaktifkan AWS IoT Greengrass untuk mengakses model yang disimpan dalam bucket Amazon S3, Anda harus memberikan AWS IoT Greengrass `read` izin untuk mengakses bucket dengan melakukan satu dari hal berikut:

- Simpan model Anda dalam bucket yang namanya berisi `greengrass`.

  Kebijakan terkelola `AWSGreengrassResourceAccessRolePolicy` mengizinkan akses ke bucket yang namanya berisi string `greengrass`. Kebijakan ini terlampir pada peran layanan Greengrass.

- Menanam kebijakan inline di peran layanan Greengrass.

  Jika nama bucket Anda tidak berisi `greengrass`, tambahkan kebijakan inline berikut untuk peran layanan. Anda dapat daftar beberapa bucket ARN.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
```

```
            "Resource": [
                "arn:aws:s3:::my-bucket-name"
            ]
        }
    ]
}
```

Untuk informasi lebih lanjut, lihat. Menanamkan kebijakan inline dalam Panduan Pengguna IAM.

Persyaratan

Persyaratan berikut berlaku untuk membuat dan menggunakan sumber daya machine learning:

- Anda harus menggunakan AWS IoT Greengrass Core v1.6 atau yang lebih baru.
- Fungsi Lambda yang ditetapkan pengguna dapat melakukan operasi `read` atau `read and write` pada sumber daya. Izin untuk operasi lain tidak tersedia. Mode kontainerisasi fungsi Lambda berafiliasi menentukan bagaimana Anda mengatur izin akses. Untuk informasi selengkapnya, lihat the section called “Mengakses sumber daya machine learning”.
- Anda harus menyediakan jalur penuh sumber daya pada sistem operasi perangkat core.
- Nama sumber daya atau ID memiliki panjang maksimum 128 karakter dan harus menggunakan pola `[a-zA-Z0-9:_-]+`.

Waktu aktif dan perpustakaan untuk inferensi ML

Anda dapat menggunakan waktu aktif ML berikut dan perpustakaan dengan AWS IoT Greengrass.

- Amazon SageMaker Waktu aktif deep learning
- Apache MXNet
- TensorFlow

Waktu aktif dan perpustakaan dapat diinstal pada platform NVIDIA Jetson TX2, Intel Atom, dan Raspberry Pi. Untuk informasi unduhan, lihat the section called “Waktu aktif dan perpustakaan machine learning yang didukung”. Anda dapat menginstalnya langsung di perangkat core Anda.

Pastikan untuk membaca informasi berikut tentang kompatibilitas dan keterbatasan.

Waktu aktif deep learning SageMaker Neo

Anda dapat menggunakan SageMaker Waktu aktif deep learning Neo untuk melakukan inferensi dengan model machine learning yang dioptimalkan diAWS IoT Greengrassperangkat. Model ini dioptimalkan menggunakan SageMaker Kompilator deep learning Neo untuk meningkatkan kecepatan prediksi inferensi machine learning. Untuk informasi lebih lanjut tentang model optimasi di SageMaker, lihat Dokumentasi SageMaker Neo.

> **Note**
>
> Saat ini, Anda dapat mengoptimalkan model machine learning menggunakan kompilator Neo deep learning di Amazon Web Services Region tertentu saja. Namun, Anda dapat menggunakan Neo deep learning runtime dengan model yang dioptimalkan di masing-masing Wilayah AWS di mana core AWS IoT Greengrass didukung. Untuk informasi lebih lanjut, lihat Cara Mengonfigurasi Inferensi Machine Learning yang Dioptimalkan.

Versioning MXNet

Apache MXNet saat ini tidak memastikan kompatibilitas maju, sehingga model yang Anda melatih menggunakan versi kerangka mungkin tidak bekerja dengan baik di versi sebelumnya dari kerangka. Untuk menghindari konflik antara tahap pelatihan model dan model, dan untuk memberikan yang konsisten end-to-end pengalaman, menggunakan versi framework MXNet yang sama di kedua tahap.

MXNet pada Raspberry P

Fungsi Greengrass Lambda yang mengakses model MXNet lokal harus mengatur variabel lingkungan berikut:

```
MXNET_ENGINE_TYPE=NativeEngine
```

Anda dapat mengatur variabel lingkungan dalam kode fungsi atau menambahkannya ke konfigurasi grup spesifik fungsi ini. Untuk contoh yang menambahkannya sebagai pengaturan konfigurasi, lihat langkah.

> **Note**
>
> Untuk penggunaan umum dari kerangka MXNet, seperti menjalankan contoh kode pihak ketiga, variabel lingkungan harus dikonfigurasi pada Raspberry Pi.

TensorFlow model melayani keterbatasan pada Raspberry Pi

Rekomendasi berikut untuk meningkatkan hasil inferensi didasarkan pada pengujian kami dengan TensorFlow Perpustakaan lengan 32-bit pada platform Raspberry Pi. Rekomendasi ini ditujukan untuk pengguna tingkat lanjut untuk referensi saja, tanpa jaminan apa pun.

- Model yang dilatih menggunakan Titik pemeriksaan format harus "dibekukan" ke format penyangga protokol sebelum disajikan. Sebagai contoh, lihat Perpustakaan model klasifikasi citra Tensorflow-slim.
- Jangan gunakan TF-estimator dan perpustakaan TF-slim baik pelatihan atau kode inferensi. Sebagai gantinya, gunakan `.pb` pola pemuatan model file yang ditunjukkan dalam contoh berikut.

```
graph = tf.Graph()
graph_def = tf.GraphDef()
graph_def.ParseFromString(pb_file.read())
with graph.as_default():
  tf.import_graph_def(graph_def)
```

> **Note**
>
> Untuk informasi selengkapnya tentang platform yang didukung untuk TensorFlow, lihatMenginstal TensorFlowdi TensorFlow dokumentasi.

Mengakses sumber daya machine learning dari fungsi Lambda

Fungsi Lambda yang ditentukan pengguna dapat mengakses sumber daya machine learning untuk menjalankan inferensi lokal pada core AWS IoT Greengrass ini. Sumber daya machine learning terdiri dari model terlatih dan artefak lainnya yang diunduh ke perangkat core.

Untuk mengizinkan fungsi Lambda mengakses sumber daya machine learning pada core, Anda harus melampirkan sumber daya ke fungsi Lambda dan menentukan izin akses. Mode kontainerisasi dari afiliasi (atau terlampir) fungsi Lambda menentukan bagaimana Anda melakukan hal ini.

Izin akses untuk sumber daya machine learning

Dimulai pada AWS IoT Greengrass Core v1.10.0, Anda dapat menentukan pemilik sumber daya untuk sumber daya machine learning. Pemilik sumber daya mewakili grup OS dan izin yang AWS IoT Greengrass menggunakan untuk mengunduh artefak sumber daya. Jika pemilik sumber daya tidak didefinisikan, artefak sumber daya yang diunduh hanya dapat diakses oleh root.

- Jika fungsi Lambda nonkontainerisasi yang mengakses sumber daya machine learning, Anda harus menentukan pemilik sumber daya karena tidak ada kontrol izin dari kontainer. Fungsi nonkontainerisasi Lambda dapat mewarisi izin pemilik sumber daya dan menggunakannya untuk mengakses sumber daya.

- Jika hanya fungsi Lambda terkontainerisasi yang mengakses sumber daya, kami sarankan Anda menggunakan izin tingkat fungsi bukannya mendefinisikan pemilik sumber daya.

Properti pemilik sumber daya

Pemilik sumber daya menentukan pemilik grup dan izin pemilik grup.

Pemilik grup. Grup ID (GID) dari grup OS Linux yang ada pada perangkat core. Izin grup ditambahkan ke proses Lambda. Secara khusus, GID ditambahkan ke ID grup tambahan dari fungsi Lambda.

Jika fungsi Lambda dalam grup Greengrass dikonfigurasi untuk jalankan sebagai grup OS yang sama dengan pemilik sumber daya untuk sumber machine learning, sumber daya harus dilampirkan pada fungsi Lambda. Jika tidak, deployment gagal karena konfigurasi ini memberikan izin implisit fungsi Lambda dapat menggunakan untuk mengakses sumber daya tanpa otorisasi AWS IoT Greengrass ini. Pemeriksaan validasi deployment dilewati jika fungsi Lambda berjalan sebagai root (UID = 0).

Kami merekomendasikan bahwa Anda menggunakan grup OS yang tidak digunakan oleh sumber daya lain, fungsi Lambda, atau file pada Greengrass core. Menggunakan grup OS bersama

memberikan melekat fungsi Lambda izin akses lebih dari yang mereka butuhkan. Jika Anda menggunakan grup OS bersama, fungsi Lambda terlampir juga harus dilampirkan ke semua sumber daya machine learning yang menggunakan grup OS bersama. Jika tidak, deployment gagal.

Izin pemilik grup. Izin hanya baca atau baca dan tulis untuk menambah proses Lambda.

Fungsi Lambda nonkontainerisasi harus mewarisi izin akses ini ke sumber daya. Fungsi Lambda terkontainerisasi dapat mewarisi izin tingkat sumber daya ini atau menentukan izin tingkat fungsi. Jika mereka menentukan izin tingkat fungsi, izin harus sama atau lebih ketat daripada izin tingkat sumber daya.

Tabel berikut menunjukkan konfigurasi izin akses yang didukung.

GGC v1.10 or later

Properti	Jika hanya fungsi Lambda terkontainerisasi yang mengakses sumber daya	Jika ada fungsi Lambda nonkontainerisasi yang mengakses sumber daya
Properti tingkat fungsi		
Izin (baca/tulis)	Diperlukan kecuali sumber daya mendefinisikan pemilik sumber daya. Jika pemilik sumber daya didefinisikan, izin tingkat fungsi harus sama atau lebih ketat daripada izin pemilik sumber daya. Jika hanya fungsi Lambda terkontainerisasi yang mengakses sumber daya, kami sarankan Anda tidak menentukan pemilik sumber daya.	Fungsi Lambda non-konta iner: Tidak didukung. Fungsi Lambda nonkontainerisasi harus mewarisi izin tingkat sumber daya. Fungsi Lambda dalam peti kemas: Opsional, tetapi harus sama atau lebih ketat daripada izin tingkat sumber daya.
Properti tingkat sumber daya		

Properti	Jika hanya fungsi Lambda terkontainerisasi yang mengakses sumber daya	Jika ada fungsi Lambda nonkontainerisasi yang mengakses sumber daya
Pemilik sumber daya	Opsional (tidak disarankan).	Wajib.
Izin (baca/tulis)	Opsional (tidak disarankan).	Wajib.

GGC v1.9 or earlier

Properti	Jika hanya fungsi Lambda terkontainerisasi yang mengakses sumber daya	Jika ada fungsi Lambda nonkontainerisasi yang mengakses sumber daya
Properti tingkat fungsi		
Izin (baca/tulis)	Wajib.	Tidak didukung.
Properti tingkat sumber daya		
Pemilik sumber daya	Tidak didukung.	Tidak didukung.
Izin (baca/tulis)	Tidak didukung.	Tidak didukung.

> **Note**
>
> Ketika Anda menggunakan API AWS IoT Greengrass untuk mengonfigurasi fungsi Lambda dan sumber daya, tingkat fungsi properti `ResourceId` juga diperlukan. Properti `ResourceId` melampirkanl sumber daya machine learning untuk fungsi Lambda.

Mendefinisikan izin akses untuk fungsi Lambda (konsol)

Di AWS IoT konsol, Anda menentukan izin akses ketika Anda mengonfigurasi sumber daya machine learning atau melampirkan satu ke fungsi Lambda.

Fungsi Lambda dalam peti kemas

Jika hanya fungsi Lambda terkontainerisasi terlampir pada sumber daya machine learning:

- Pilih Tidak ada grup sistem sebagai pemilik sumber daya untuk sumber daya pembelajaran mesin. Ini adalah pengaturan yang direkomendasikan ketika hanya fungsi Lambda terkontainerisasi yang mengakses sumber daya machine learning. Jika tidak, Anda mungkin memberikan fungsi Lambda yang dilampirkan izin akses lebih dari yang mereka butuhkan.

Fungsi Lambda nonkontainerisasi (memerlukankan GC v1.10 atau lebih baru)

Jika ada fungsi Lambda nonkontainerisasi terlampir pada sumber daya machine learning:

- Tentukan ID grup Sistem (GID) yang akan digunakan sebagai pemilik sumber daya untuk sumber daya pembelajaran mesin. Pilih Tentukan grup sistem dan izin dan masukkan GID. Anda dapat menggunakan `getent group` perintah pada perangkat inti Anda untuk mencari ID grup sistem.

- Pilih Akses hanya-baca atau akses Baca dan tulis untuk izin grup Sistem.

Mendefinisikan izin akses untuk fungsi Lambda (API)

Di AWS IoT Greengrass API, Anda menentukan izin untuk sumber daya machine learning di `ResourceAccessPolicy` Properti untuk fungsi Lambda atau `OwnerSetting` properti untuk sumber daya.

Fungsi Lambda dalam peti kemas

Jika hanya fungsi Lambda terkontainerisasi terlampir pada sumber daya machine learning:

- Untuk fungsi Lambda terkontainerisasi, tentukan izin akses di properti `Permission` dari properti `ResourceAccessPolicies` ini. Sebagai contoh:

```
"Functions": [
    {
        "Id": "my-containerized-function",
        "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:function-
name:alias-or-version",
```

```
        "FunctionConfiguration": {
            "Environment": {
                "ResourceAccessPolicies": [
                    {
                        "ResourceId": "my-resource-id",
                        "Permission": "ro-or-rw"
                    }
                ]
            },
            "MemorySize": 512,
            "Pinned": true,
            "Timeout": 5
        }
    }
]
```

- Untuk sumber daya machine learning, abaikan properti `OwnerSetting` ini. Sebagai contoh:

```
"Resources": [
    {
        "Id": "my-resource-id",
        "Name": "my-resource-name",
        "ResourceDataContainer": {
            "S3MachineLearningModelResourceData": {
                "DestinationPath": "/local-destination-path",
                "S3Uri": "s3://uri-to-resource-package"
            }
        }
    }
]
```

  Ini adalah konfigurasi yang direkomendasikan ketika hanya fungsi Lambda terkontainerisasi yang mengakses sumber daya machine learning. Jika tidak, Anda mungkin memberikan fungsi Lambda yang dilampirkan izin akses lebih dari yang mereka butuhkan.

Fungsi Lambda nonkontainerisasi (memerlukankan GC v1.10 atau lebih baru)

Jika ada fungsi Lambda nonkontainerisasi terlampir pada sumber daya machine learning:

- Untuk fungsi Lambda nonkontainerisasi, abaikan properti `Permission` di `ResourceAccessPolicies`. Konfigurasi ini diperlukan dan memungkinkan fungsi untuk mewarisi izin tingkat sumber daya. Sebagai contoh:

```
"Functions": [
    {
        "Id": "my-non-containerized-function",
        "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:function-
name:alias-or-version",
        "FunctionConfiguration": {
            "Environment": {
                "Execution": {
                    "IsolationMode": "NoContainer",
                },
                "ResourceAccessPolicies": [
                    {
                        "ResourceId": "my-resource-id"
                    }
                ]
            },
            "Pinned": true,
            "Timeout": 5
        }
    }
]
```

- Untuk fungsi Lambda terkontainerisasi yang juga mengakses sumber daya machine learning, abaikan properti `Permission` di `ResourceAccessPolicies` atau tentukan izin yang sama atau lebih ketat sebagai izin tingkat sumber daya. Sebagai contoh:

```
"Functions": [
    {
        "Id": "my-containerized-function",
        "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:function-
name:alias-or-version",
        "FunctionConfiguration": {
            "Environment": {
                "ResourceAccessPolicies": [
                    {
                        "ResourceId": "my-resource-id",
                        "Permission": "ro-or-rw" // Optional, but cannot exceed
 the GroupPermission defined for the resource.
                    }
                ]
            },
            "MemorySize": 512,
```

```
            "Pinned": true,
            "Timeout": 5
        }
    }
]
```

- Untuk sumber daya machine learning, tentukan `OwnerSetting` properti, termasuk anak `GroupOwner` dan `GroupPermission` properti. Sebagai contoh:

```
"Resources": [
    {
        "Id": "my-resource-id",
        "Name": "my-resource-name",
        "ResourceDataContainer": {
            "S3MachineLearningModelResourceData": {
                "DestinationPath": "/local-destination-path",
                "S3Uri": "s3://uri-to-resource-package",
                "OwnerSetting": {
                    "GroupOwner": "os-group-id",
                    "GroupPermission": "ro-or-rw"
                }
            }
        }
    }
]
```

Mengakses sumber daya machine learning dari kode fungsi Lambda

Fungsi Lambda yang ditentukan pengguna menggunakan antarmuka OS platform spesifik untuk mengakses sumber daya machine learning pada perangkat core.

GGC v1.10 or later

Untuk fungsi Lambda terkontainerisasi, sumber daya dipasang di dalam kontainer Greengrass dan tersedia di jalur tujuan lokal yang didefinisikan untuk sumber daya. Untuk fungsi Lambda nonkontainerisasi, sumber daya adalah symlinked ke direktori kerja Lambda-spesifik dan diteruskan ke `AWS_GG_RESOURCE_PREFIX` variabel lingkungan dalam proses Lambda.

Untuk mendapatkan jalur ke artefak yang diunduh dari sumber daya machine learning, fungsi Lambda menambahkan `AWS_GG_RESOURCE_PREFIX` ke jalur tujuan lokal yang ditetapkan untuk

sumber daya. Untuk fungsi Lambda terkontainerisasi, nilai yang dikembalikan adalah garis miring tunggal (/).

```
resourcePath = os.getenv("AWS_GG_RESOURCE_PREFIX") + "/destination-path"
with open(resourcePath, 'r') as f:
    # load_model(f)
```

GGC v1.9 or earlier

Artefak yang diunduh dari sumber machine learning terletak di jalur tujuan lokal yang ditentukan untuk sumber daya. Hanya fungsi Lambda terkontainerisasi yang dapat mengakses sumber daya machine learning di Core AWS IoT Greengrass v1.9 dan sebelumnya.

```
resourcePath = "/local-destination-path"
with open(resourcePath, 'r') as f:
    # load_model(f)
```

Implementasi pemuatan model Anda tergantung pada perpustakaan ML Anda.

Memecahkan masalah

Gunakan informasi berikut untuk membantu memecahkan masalah dengan mengakses sumber daya machine learning.

Topik

- InvalidML ModelOwner - GroupOwnerSetting disediakan dalam sumber daya model ML, tetapi GroupOwner atau GroupPermission tidak ada
- NoContainer fungsi tidak dapat mengonfigurasi izin saat melampirkan sumber daya Machine Learning. <function-arn>mengacu pada sumber daya Machine Learnin <resource-id>dengan izin <ro/rw> dalam kebijakan akses sumber daya.
- Fungsi <function-arn>mengacu pada sumber daya Machine Learning <resource-id>dengan izin yang hilang di keduanya ResourceAccessPolicy dan sumber daya OwnerSetting.
- Fungsi <function-arn>mengacu pada sumber daya Machine Learning <resource-id>dengan izin\ "rw\", sedangkan pengaturan pemilik sumber daya GroupPermission hanya mengizinkan\ "ro\".
- NoContainer Fungsi <function-arn>mengacu pada sumber daya jalur tujuan bersarang.
- Lambda <function-arn> mendapatkan akses ke sumber daya <resource-id> dengan berbagi id pemilik grup yang sama

InvalidML ModelOwner - GroupOwnerSetting disediakan dalam sumber daya model ML, tetapi GroupOwner atau GroupPermission tidak ada

Solusi: Anda menerima kesalahan ini jika sumber pembelajaran mesin berisi ResourceDownloadOwnerSettingobjek tetapi diperlukan `GroupOwner` atau `GroupPermission` properti tidak ditentukan. Untuk mengatasi masalah ini, tentukan properti yang hilang.

NoContainer fungsi tidak dapat mengonfigurasi izin saat melampirkan sumber daya Machine Learning. <function-arn>mengacu pada sumber daya Machine Learnin <resource-id>dengan izin <ro/rw> dalam kebijakan akses sumber daya.

Solusi: Anda menerima error ini jika fungsi Lambda non-containerized menentukan tingkat fungsi izin untuk sumber daya machine learning. Fungsi non-wadah harus mewarisi izin dari izin pemilik sumber daya yang ditetapkan pada sumber daya machine learning. Untuk mengatasi masalah ini, pilih untuk mewarisi izin pemilik sumber daya (konsol) atau menghapus izin dari fungsi Lambda sumber daya kebijakan akses (API).

Fungsi <function-arn>mengacu pada sumber daya Machine Learning <resource-id>dengan izin yang hilang di keduanya ResourceAccessPolicy dan sumber daya OwnerSetting.

Solusi: Anda menerima kesa ini jika izin untuk sumber daya machine learning tidak dikonfigurasi untuk fungsi Lambda terlampir atau sumber daya. Untuk mengatasi masalah ini, konfigurasikan izin di ResourceAccessPolicyproperti untuk fungsi Lambda atau properti untuk OwnerSettingsumber daya.

Fungsi <function-arn>mengacu pada sumber daya Machine Learning <resource-id>dengan izin\ "rw\”, sedangkan pengaturan pemilik sumber daya GroupPermission hanya mengizinkan\ "ro\”.

Solusi: Anda menerima error ini jika izin akses yang ditetapkan untuk fungsi Lambda terlampir melebihi izin pemilik sumber daya yang ditetapkan untuk sumber daya machine learning. Untuk mengatasi masalah ini, tetapkan izin yang lebih ketat untuk fungsi Lambda atau kurang membatasi izin untuk pemilik sumber daya.

NoContainer Fungsi <function-arn>mengacu pada sumber daya jalur tujuan bersarang.

Solusi: Anda menerima error ini jika beberapa sumber daya machine learning yang terlampir pada fungsi Lambda non-containerized menggunakan lintasan tujuan yang sama atau lintasan tujuan bersarang. Untuk mengatasi masalah ini, tentukan jalur tujuan terpisah untuk sumber daya.

Lambda <function-arn> mendapatkan akses ke sumber daya <resource-id> dengan berbagi id pemilik grup yang sama

Solusi: Anda menerima keslaahan ini di `runtime.log` jika kelompok OS yang sama ditentukan sebagai fungsi Lambda Jalankan sebagai identitas dan pemilik sumber daya untuk sumber daya machine learning, tetapi sumber daya tidak terlampir pada fungsi Lambda. Konfigurasi ini memberikan fungsi Lambda izin implisit yang dapat digunakan untuk mengakses sumber daya tanpa AWS IoT Greengrass otorisasi.

Untuk mengatasi masalah ini, gunakan grup OS yang berbeda untuk salah satu properti atau melampirkan sumber daya machine learning untuk fungsi Lambda.

Lihat juga

- Tampilkan inferensi machine learning
- the section called “Cara mengonfigurasi inferensi machine learning”
- the section called “Cara mengonfigurasi kesimpulan machine learning yang dioptimalkan”
- Referensi API AWS IoT Greengrass Version 1

Cara mengonfigurasi inferensi machine learning menggunakan AWS Management Console

Untuk mengikuti langkah-langkah di tutorial ini, Anda perlu AWS IoT Greengrass Core v1.10 atau yang lebih baru.

Anda dapat melakukan machine learning (ML) inferensi lokal pada perangkat core Greengrass menggunakan data yang dihasilkan secara lokal. Untuk informasi, termasuk persyaratan dan kendala, lihat Tampilkan inferensi machine learning.

Tutorial ini menjelaskan cara menggunakan AWS Management Console untuk mengonfigurasi grup Greengrass agar menjalankan aplikasi inferensi Lambda yang mengenali citra dari kamera secara lokal, tanpa mengirim data ke cloud. Aplikasi inferensi mengakses modul kamera pada Raspberry Pi dan menjalankan inferensi menggunakan sumber terbukaSqueezeNetModel.

Tutorial ini berisi langkah-langkah tingkat tinggi berikut:

1. Mengonfigurasi Raspberry Pi
2. Instal kerangka kerja MXNet
3. Buat paket model
4. Buat dan publikasikan fungsi Lambda
5. Menambahkan fungsi Lambda ke grup
6. Menambahkan sumber daya ke grup
7. Menambahkan langganan ke grup
8. Men-deploy grup
9. Tes aplikasi

Prasyarat

Untuk menyelesaikan tutorial ini, Anda memerlukan:

- Raspberry Pi 4 Model B, atau Raspberry Pi 3 Model B/B+, diatur dan dikonfigurasi untuk digunakan dengan AWS IoT Greengrass. Untuk mengatur Raspberry Pi Anda dengan AWS IoT Greengrass, jalankan Pengaturan Perangkat Greengrass tersebut, atau pastikan bahwa Anda telah menyelesaikan Modul 1 dan Modul 2 dari Mulai menggunakan AWS IoT Greengrass.

  > **Note**
  >
  > Raspberry Pi mungkin memerlukan Catu daya 2.5A untuk menjalankan kerangka kerja deep learning yang biasanya digunakan untuk klasifikasi citra. Catu daya dengan peringkat lebih rendah dapat menyebabkan perangkat melakukan reboot.

- Modul Kamera Raspberry Pi V2 - 8 megapiksel, 1080p. Untuk informasi tentang cara mengatur kamera, lihat Menghubungkan kamera dalam dokumentasi Raspberry Pi.
- Sebuah grup Greengrass dan core Greengrass. Untuk informasi tentang cara membuat grup Greengrass atau core, lihat Mulai menggunakan AWS IoT Greengrass.

> **Note**
>
> Tutorial ini menggunakan Raspberry Pi, AWS IoT Greengrass support platform lain, seperti Intel Atom dan NVIDIA Jetson TX2. Dalam contoh untuk Jetson TX2, Anda dapat menggunakan citra statis, bukan citra yang dialirkan dari kamera. Jika menggunakan contoh Jetson TX2, Anda mungkin perlu menginstal Python 3.6 bukan Python 3.7. Untuk informasi tentang mengonfigurasi perangkat Anda, bahwa Anda dapat menginstal perangkat lunak AWS IoT Greengrass core, lihat the section called “Mengatur perangkat lain”.
> Untuk platform pihak ketiga di mana AWS IoT Greengrass tidak mendukung, Anda harus menjalankan fungsi Lambda Anda dalam mode nonkontainerisasi. Untuk menjalankan dalam mode nonkontainerisasi, Anda harus menjalankan fungsi Lambda Anda sebagai root. Untuk informasi lebih lanjut, lihat the section called “Pertimbangan ketika memilih fungsi Lambda kontainerisasi” dan the section called “Mengatur identitas akses default untuk fungsi Lambda dalam grup”.

Langkah 1: Mengonfigurasi Raspberry Pi

Pada langkah ini, instal pembaruan untuk sistem operasi Raspbian, instal perangkat lunak modul kamera dan Python dependensi, dan mengaktifkan antarmuka kamera.

Jalankan perintah berikut di terminal Raspberry Pi Anda.

1. Menginstal pembaruan untuk Raspbian.

   ```
   sudo apt-get update
   sudo apt-get dist-upgrade
   ```

2. Instal antarmuka `picamera` untuk modul kamera dan pustaka Python lain yang diperlukan untuk tutorial ini.

   ```
   sudo apt-get install -y python3-dev python3-setuptools python3-pip python3-picamera
   ```

   Memvalidasi instalasi:

   - Pastikan bahwa instalasi Python 3.7 anda termasuk pip.

     ```
     python3 -m pip
     ```

Jika pip tidak terpasang, unduh dari Situs web pip tersebut, kemudian jalankan perintah berikut.

```
python3 get-pip.py
```

- Pastikan bahwa versi Python Anda adalah 3.7 atau lebih tinggi.

```
python3 --version
```

Jika output mencantumkan versi sebelumnya, jalankan perintah berikut.

```
sudo apt-get install -y python3.7-dev
```

- Pastikan bahwa Setuptools dan Picamera berhasil diinstal.

```
sudo -u ggc_user bash -c 'python3 -c "import setuptools"'
sudo -u ggc_user bash -c 'python3 -c "import picamera"'
```

Jika output tidak mengandung error, validasi berhasil.

> **Note**
>
> Jika Python yang dapat dieksekusi diinstal pada perangkat anda adalah `python3.7`, gunakan `python3.7` daripada `python3` untuk perintah di tutorial ini. Pastikan bahwa instalasi pip Anda memetakan versi `python3.7` atau `python3` yang tepat untuk menghindari kesalahan dependensi.

3. Reboot Raspberry Pi.

```
sudo reboot
```

4. Buka alat konfigurasi Raspberry Pi.

```
sudo raspi-config
```

5. Gunakan tombol panah untuk membuka Opsi Antarmuka dan mengaktifkan antarmuka kamera. Jika diminta, izinkan perangkat melakukan reboot.
6. Gunakan perintah berikut untuk menguji pengaturan kamera.

```
raspistill -v -o test.jpg
```

Hal ini akan membuka jendela pratinjau pada Raspberry Pi, menyimpan gambar bernama `test.jpg` ke direktori Anda saat ini, dan menampilkan informasi tentang kamera di terminal Raspberry Pi.

Langkah 2: Instal kerangka kerja MXNet

Dalam langkah ini, menginstal perpustakaan MXNet pada Raspberry Pi Anda.

1. Masuk ke Raspberry Pi Anda dari jarak jauh.

   ```
   ssh pi@your-device-ip-address
   ```

2. Buka dokumentasi MXNet, buka Menginstal MXNet, lalu ikuti petunjuk untuk menginstal MXNet di perangkat.

   > **Note**
   >
   > Kami menyarankan untuk menginstal versi 1.5.0 dan membangun MXNet dari sumber untuk tutorial ini agar menghindari konflik perangkat.

3. Setelah Anda menginstal MXNet, validasi konfigurasi berikut:

   - Pastikan akun sistem `ggc_user` dapat menggunakan kerangka kerja MXNet.

     ```
     sudo -u ggc_user bash -c 'python3 -c "import mxnet"'
     ```

   - Pastikan NumPy diinstal.

     ```
     sudo -u ggc_user bash -c 'python3 -c "import numpy"'
     ```

Langkah 3: Membuat paket model MXNet

Pada langkah ini, buat paket model yang memuat contoh model MXNet yang terlatih untuk mengunggah ke Amazon Simple Storage Service (Amazon S3). AWS IoT Greengrass dapat menggunakan paket model dari Amazon S3, asalkan Anda menggunakan format tar.gz atau zip.

1. Pada komputer anda, unduh sampel MXNet untuk Raspberry Pi dari the section called “Sampel machine learning”.
2. Unzip file `mxnet-py3-armv7l.tar.gz` yang diunduh.
3. Buka direktori `squeezenet` tersebut.

   ```
   cd path-to-downloaded-sample/mxnet-py3-armv7l/models/squeezenet
   ```

   File `squeezenet.zip` dalam direktori ini adalah paket model Anda. Ini berisi SqueezeNet artefak model sumber terbuka untuk model klasifikasi gambar. Kemudian, Anda mengunggah paket model ini ke Amazon S3.

Langkah 4: Buat dan publikasikan fungsi Lambda

Pada langkah ini, buat paket deployment fungsi Lambda dan fungsi Lambda. Kemudian, publikasikan versi fungsi dan buat alias.

Pertama, buat paket deployment fungsi Lambda.

1. Di komputer Anda, navigasikan ke direktori `examples` dalam paket contoh yang Anda unzip di the section called “Buat paket model”.

   ```
   cd path-to-downloaded-sample/mxnet-py3-armv7l/examples
   ```

   Direktori `examples` berisi kode fungsi dan dependensi.

   - `greengrassObjectClassification.py` adalah kode inferensi yang digunakan dalam tutorial ini. Anda dapat menggunakan kode ini sebagai templat untuk membuat fungsi inferensi Anda sendiri.
   - `greengrasssdk` adalah versi 1.5.0 dari AWS IoT Greengrass Core SDK for Python.

   > **Note**
   >
   > Jika versi baru tersedia, Anda dapat mengunduh dan membarui versi SDK dalam paket deployment Anda. Untuk informasi selengkapnya, lihatAWS IoT GreengrassCore SDK for Pythondi atas GitHub.

2. Kompres isi `examples` ke dalam sebuah file bernama `greengrassObjectClassification.zip`. Ini paket deployment Anda.

```
zip -r greengrassObjectClassification.zip .
```

> **Note**
>
> Pastikan `.py` dan dependensi berada di root direktori.

Selanjutnya, Buat fungsi Lambda.

3. DariAWS IoTkonsol, pilihFungsidanBuat fungsi.
4. Pilih Tulis dari awal dan gunakan nilaiinilai berikut untuk membuat fungsi Anda:

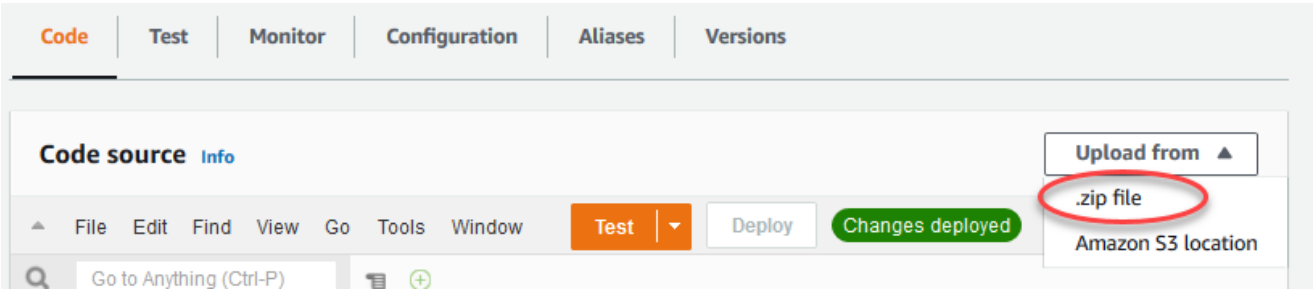
   - Untuk Nama fungsi, masukkan **`greengrassObjectClassification`**.
   - Untuk Waktu pengoperasian, pilih Python 3.7.

   Untuk Izin, pertahankan pengaturan default. Hal ini menciptakan peran eksekusi yang memberikan izin Lambda basic. Peran ini tidak digunakan oleh AWS IoT Greengrass.
5. Pilih Buat fungsi.

Sekarang, unggah paket deployment fungsi Lambda Anda dan daftarkan handler.

6. Pilih fungsi Lambda Anda dan unggah paket deployment fungsi Lambda Anda.

   a. Pada tab Kode ini, di bawah Sumber kode, pilih Unggah dari. Dari dropdown, pilih file .zip.



b. Pilih Unggah, lalu pilih paket deployment `greengrassObjectClassification.zip` Anda. Lalu, pilih Simpan.

c. Pada tab Kode fungsi, di bawah Pengaturan waktu aktif, pilih Edit, dan kemudian masukkan nilai-nilai berikut.

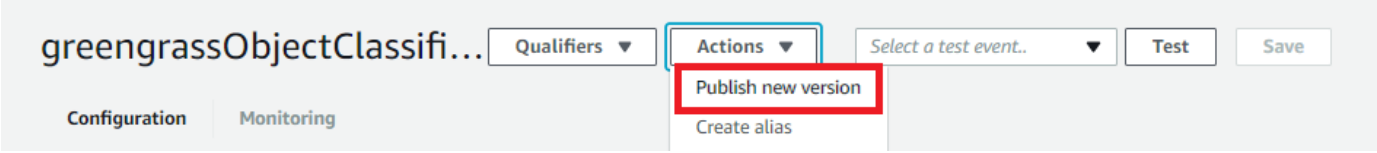
- Untuk waktu aktif, pilih Python 3.7.
- Untuk Handler, masukkan **`greengrassObjectClassification.function_handler`**.

Pilih Save (Simpan).

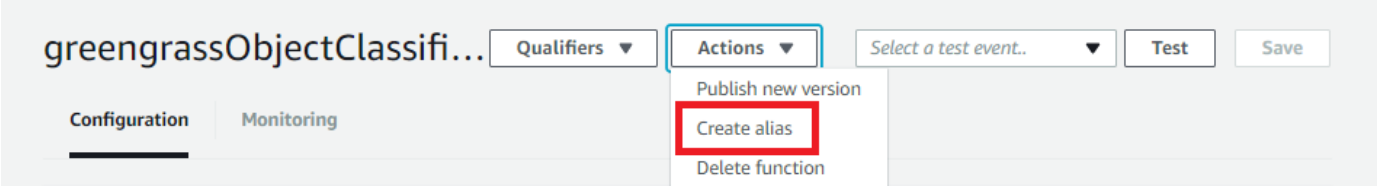
Berikutnya, mempublikasikan versi pertama fungsi Lambda Anda. Kemudian, buat alias untuk versi.

> **Note**
>
> Grup Greengrass dapat mereferensi fungsi Lambda dengan alias (direkomendasikan) atau dengan versi. Menggunakan alias membuatnya lebih mudah untuk mengelola pembaruan kode karena Anda tidak perlu mengubah tabel langganan atau definisi grup ketika kode fungsi diperbarui. Sebaliknya, Anda hanya mengarahkan alias ke versi fungsi baru.

7. Dari menu Tindakan tersebut, pilih Publikasikan versi baru.



8. Untuk Versi Deskripsi, masukkan **`First version`**, lalu pilih Publikasikan.
9. PadagreengrassObjectClassification: 1halaman konfigurasi, dariTindakanmenu, pilihMembuat alias.



10. Pada halaman Buat alias baru tersebut, gunakan nilai-nilai berikut:

- Untuk Nama, masukkan **`mlTest`**.
- Untuk UID, masukkan **`1`**.

> **Note**
>
> AWS IoT Greengrass tidak support alias Lambda untuk versi $TERBARU ini.

11. Pilih Save (Simpan).

Sekarang, tambahkan fungsi Lambda ke grup Greengrass Anda.

Langkah 5: Tambahkan fungsi Lambda ke grup Greengrass

Pada langkah ini, menambahkan fungsi Lambda ke grup dan kemudian mengonfigurasi siklus hidup dan lingkungan variabel.

Pertama, tambahkan fungsi Lambda ke grup Greengrass Anda.

1. DiAWS IoTpanel navigasi konsol, di bawahKelola, perluasPerangkat Greengrass, dan kemudian pilihGrup (V1).
2. Dari halaman konfigurasi grup, pilihFungsi LambdaTab.
3. Di bawahFungsi Lambda sayabagian, pilihTambahkan.
4. UntukFungsi Lambda, PilihgreengrassObjectClassification.
5. UntukVersi fungsi Lambda, PilihAlias: MLTest.

Selanjutnya, konfigurasikan variabel siklus hidup dan lingkungan dari fungsi Lambda.

6. PadaKonfigurasi fungsi Lambdabagian, buat pembaruan berikut.

> **Note**
>
> Kami merekomendasikan Anda menjalankan fungsi Lambda Anda tanpa kontainerisasi kecuali kasus bisnis Anda memerlukannya. Hal ini membantu mengaktifkan akses ke perangkat GPU dan kamera Anda tanpa mengonfigurasi sumber daya perangkat. Jika

Anda menjalankan tanpa kontainerisasi, Anda juga harus memberikan akses root ke Fungsi Lambda AWS IoT Greengrass Anda.

a. Untuk berjalan tanpa kontainerisasi:

- UntukPengguna dan grup sistem, Pilih**Another user ID/group ID**. UntukID pengguna SistemENTER**0**. UntukID Grup SistemENTER**0**.

  Hal ini memungkinkan fungsi Lambda Anda untuk berjalan sebagai root. Untuk informasi lebih lanjut untuk menjalankan sebagai root, lihat the section called “Mengatur identitas akses default untuk fungsi Lambda dalam grup”.

  > **Tip**
  >
  > Anda juga harus memperbarui file `config.json` untuk memberikan akses root ke fungsi Lambda Anda. Untuk prosedur ini, lihat the section called “Menjalankan fungsi Lambda sebagai root”.

- UntukFungsi Lambda kontainerisasi, PilihTanpa kontainer.

  Untuk informasi lebih lanjut untuk berjalan tanpa konainerisasi, lihat the section called “Pertimbangan ketika memilih fungsi Lambda kontainerisasi”.

- Untuk Timeout, masukkan **10 seconds**.
- UntukDipinilih, PilihBenar.

  Untuk informasi selengkapnya, lihat the section called “Konfigurasi siklus hidup”.

b. Untuk menjalankan dalam mode kontainer sebagai gantinya:

> **Note**
>
> Kami tidak merekomendasikan untuk dijalankan dalam mode kontainer kecuali jika kasus bisnis Anda memerlukannya.

- UntukPengguna dan grup sistem, PilihGunakan default grup.
- UntukFungsi Lambda kontainerisasi, PilihGunakan default grup.
- Untuk Batas memori, masukkan **96 MB**.

- Untuk Timeout, masukkan **`10 seconds`**.
- UntukDipinilih, PilihBenar.

  Untuk informasi selengkapnya, lihat the section called “Konfigurasi siklus hidup”.

7. Di bawah Lingkungan variabel, buat pasangan kunci-nilai. Pasangan kunci-nilai diperlukan oleh fungsi yang berinteraksi dengan model MXNet pada Raspberry Pi.

   Untuk kuncinya, gunakan MXNET_ENGINE_TYPE. Untuk nilai, gunakanNaiveEngine.

   > **Note**
   >
   > Dalam fungsi Lambda yang ditetapkan pengguna milik Anda, Anda secara opsional dapat mengatur variabel lingkungan dalam kode fungsi Anda.

8. Menjaga nilai default untuk semua properti lain dan memilihMenambahkan fungsi Lambda.

Langkah 6: Menambahkan sumber daya ke grup Greengrass

Pada langkah ini, membuat sumber daya untuk modul kamera dan model inferensi ML dan afiliasi sumber daya dengan fungsi Lambda. Hal ini memungkinkan fungsi Lambda untuk mengakses sumber daya pada perangkat core.

> **Note**
>
> Jika Anda berjalan dalam mode nonkontainerisasi, AWS IoT Greengrass dapat mengakses GPU dan kamera dari perangkat Anda tanpa mengonfigurasi sumber daya perangkat ini.

Pertama, buat dua sumber daya perangkat lokal untuk kamera: satu untuk memori berbagi dan satu untuk antarmuka perangkat. Untuk informasi lebih lanjut tentang akses sumber daya lokal, lihat Akses sumber daya lokal dengan fungsi dan konektor Lambda.

1. Pada halaman konfigurasi grup, pilihSumber dayaTab.
2. DiSumber daya lokalbagian, pilihMenambahkan sumber daya lokal.
3. PadaMenambahkan sumber daya lokalhalaman, gunakan nilai-nilai berikut:

   - Untuk Nama sumber daya, masukkan **`videoCoreSharedMemory`**.

- Untuk Jenis sumber daya, pilih Perangkat.
- UntukJalur perangkat lokalENTER**`/dev/vcsm`**.

  Jalur perangkat adalah jalur absolut lokal sumber daya perangkat. Jalur ini hanya dapat merujuk ke perangkat karakter atau memblokir perangkat di bawah `/dev`.
- UntukPemilik grup sistem dan izin akses file, PilihSecara otomatis menambahkan izin sistem file dari grup sistem yang memiliki sumber daya.

  YangPemilik grup sistem dan izin akses fileOpsi memungkinkan Anda memberikan izin akses file tambahan untuk proses Lambda. Untuk informasi selengkapnya, lihat Izin akses file pemilik grup.

4. Selanjutnya, Anda menambahkan sumber daya perangkat lokal untuk antarmuka kamera.
5. PilihMenambahkan sumber daya lokal.
6. PadaMenambahkan sumber daya lokalhalaman, gunakan nilai-nilai berikut:

   - Untuk Nama sumber daya, masukkan **`videoCoreInterface`**.
   - Untuk Jenis sumber daya, pilih Perangkat.
   - UntukJalur perangkat lokalENTER**`/dev/vchiq`**.
   - UntukPemilik grup sistem dan izin akses file, PilihSecara otomatis menambahkan izin sistem file dari grup sistem yang memiliki sumber daya.
7. Di bagian bawah halaman, pilihTambahkan sumber daya.

Sekarang, tambahkan model inferensi sebagai sumber daya machine learning. Langkah ini mencakup mengunggah paket model `squeezenet.zip` ke Amazon S3.

1. PadaSumber dayatab untuk grup Anda, di bawahMachine Learningbagian, pilihMenambahkan sumber daya machine learning.
2. PadaMenambahkan sumber daya pembelajaran mesinhalamanNama sumber dayaENTER**`squeezenet_model`**.
3. UntukSumber model, PilihGunakan model yang disimpan di S3, seperti model yang dioptimalkan melalui Deep Learning Compiler.
4. UntukJENIS S3, masukkan jalur tempat bucket S3 disimpan.
5. Pilih Jelajahi S3. Ini membuka tab baru ke konsol Amazon S3.

6. Pada konsol Amazon S3, unggah file `squeezenet.zip` ke bucket S3. Untuk informasi, lihatBagaimana cara mengunggah file dan folder ke S3 Bucket?di dalamPanduan Pengguna Amazon Simple Storage Service.

   > **Note**
   >
   > Agar bucket S3 dapat diakses, nama bucket Anda harus berisi string **`greengrass`** dan bucket harus berada di wilayah yang sama dengan yang Anda gunakan pada AWS IoT Greengrass. Pilih nama yang unik (seperti **`greengrass-bucket-`*`user-id`*`-`*`epoch-time`**). Jangan gunakan periode (`.`) dalam nama bucket.

7. Pada tab konsol AWS IoT Greengrass tersebut, cari dan pilih bucket S3 Anda. Temukan file `squeezenet.zip` yang Anda unggah, dan pilih Pilihan. Anda mungkin harus memilih Refresh untuk memperbarui daftar bucket dan file yang tersedia.
8. Untuk Jalur tujuan, masukkan **`/greengrass-machine-learning/mxnet/squeezenet`**.

   Hal ini adalah tujuan untuk model lokal di Lambda waktu aktif namespace. Saat Anda men-deploy grup, AWS IoT Greengrass mengambil paket model sumber dan kemudian ekstrak isinya ke direktori tertentu. Contoh fungsi Lambda untuk tutorial ini sudah dikonfigurasi untuk menggunakan jalur ini (dalam variabel `model_path` tersebut).
9. Di bawahPemilik grup sistem dan izin akses file, PilihTidak ada kelompok sistem.
10. PilihTambahkan sumber daya.

Menggunakan SageMaker model terlatih

Tutorial ini menggunakan model yang disimpan di Amazon S3, tetapi Anda dapat dengan mudah menggunakannya SageMaker model juga. YangAWS IoT Greengrasskonsol telah built-in SageMaker integrasi, sehingga Anda tidak perlu secara manual mengunggah model ini ke Amazon S3. Untuk persyaratan dan batasan untuk menggunakan SageMaker model, lihatthe section called “Sumber model yang didukung”.

Untuk menggunakan SageMaker Model:

- UntukSumber model, PilihGunakan model yang dilatihAWS SageMaker, lalu pilih nama tugas pelatihan model.
- UntukJalur tujuan, masukkan jalur ke direktori dimana fungsi Lambda anda mencari modelnya.

Langkah 7: Menambahkan langganan ke grup Greengrass

Pada langkah ini, tambahkan langganan ke grup. Langganan ini memungkinkan fungsi Lambda untuk mengirim hasil prediksi ke AWS IoT dengan menerbitkan topik MQTT.

1. Pada halaman konfigurasi grup, pilihLangganantab, dan kemudian pilihMenambahkan langganan.
2. PadaDetail langgananhalaman, mengkonfigurasi sumber dan target, sebagai berikut:
   a. MasukJenis Sumber, PilihFungsi Lambda, dan kemudian pilihgreengrassObjectClassification.
   b. MasukJenis target, PilihLayanan, dan kemudian pilihIoT Cloud.
3. MasukFilter topikENTER**`hello/world`**, dan kemudian pilihBuat langganan.

Langkah 8: Men-deploy grup Greengrass

Pada langkah ini, men-deploy versi definisi grup untuk perangkat core Greengrass. Definisi berisi fungsi Lambda, sumber daya, dan konfigurasi langganan yang ditambahkan.

1. Pastikan bahwa core AWS IoT Greengrass sedang berjalan. Jalankan perintah berikut di terminal Raspberry Pi Anda, sesuai kebutuhan.
   a. Untuk memeriksa apakah daemon sedang berjalan:

   ```
   ps aux | grep -E 'greengrass.*daemon'
   ```

   Jika outputnya berisi entri `root` untuk `/greengrass/ggc/packages/1.11.6/bin/daemon`, maka daemon sedang berjalan.

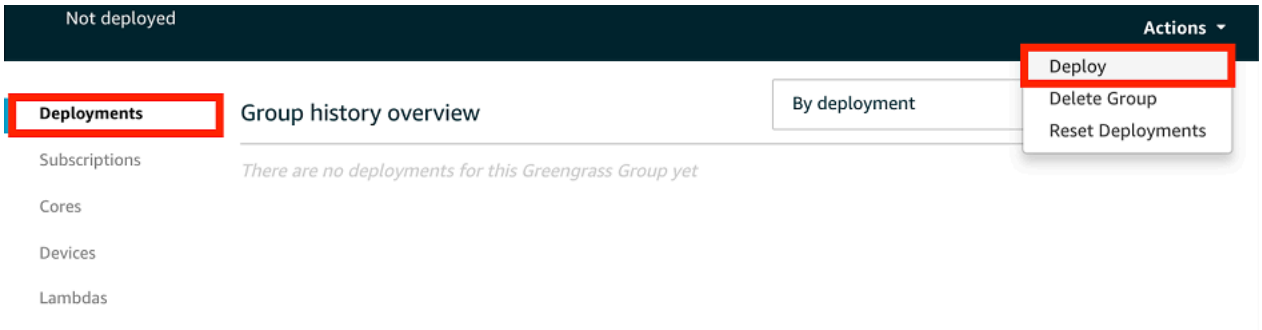
   > **Note**
   > Versi di jalur tergantung pada versi perangkat lunak AWS IoT Greengrass core yang diinstal pada perangkat core Anda.

   b. Untuk memulai daemon:

   ```
   cd /greengrass/ggc/core/
   ```

```
sudo ./greengrassd start
```

2. Pada halaman konfigurasi grup, pilihDeploy.



3. DiFungsi Lambdatab, di bawahFungsi Lambda sistembagian, pilihDetektor IPdan pilihlahedit.
4. DiEdit pengaturan detektor IPkotak dialog, pilihSecara otomatis mendeteksi dan mengganti titik akhir broker MQTT.
5. Pilih Save (Simpan).

   Hal ini mengaktifkan perangkat untuk secara otomatis memperoleh informasi konektivitas untuk core, seperti alamat IP, DNS, dan nomor port. Deteksi otomatis direkomendasikan, namun AWS IoT Greengrass juga support titik akhir yang ditentukan secara manual. Anda hanya diminta untuk metode penemuan pertama kalinya bahwa grup di-deploy.

   > **Note**
   >
   > Jika diminta, berikan izin untuk membuat Peran layanan Greengrass dan kaitkan dengan Akun AWS Anda pada Wilayah AWS. Peran ini memungkinkan AWS IoT Greengrass untuk mengakses sumber daya Anda di layanan AWS ini.

   Halaman Deployment menampilkan timestamp deployment, ID versi, dan status. Setelah selesai, status yang ditampilkan untuk deployment seharusnyaCompleted (Lengkap).

   Untuk informasi lebih lanjut tentang deployment, lihat Men-deploy AWS IoT Greengrass grup. Untuk bantuan penyelesaian masalah, lihat Memecahkan masalah.

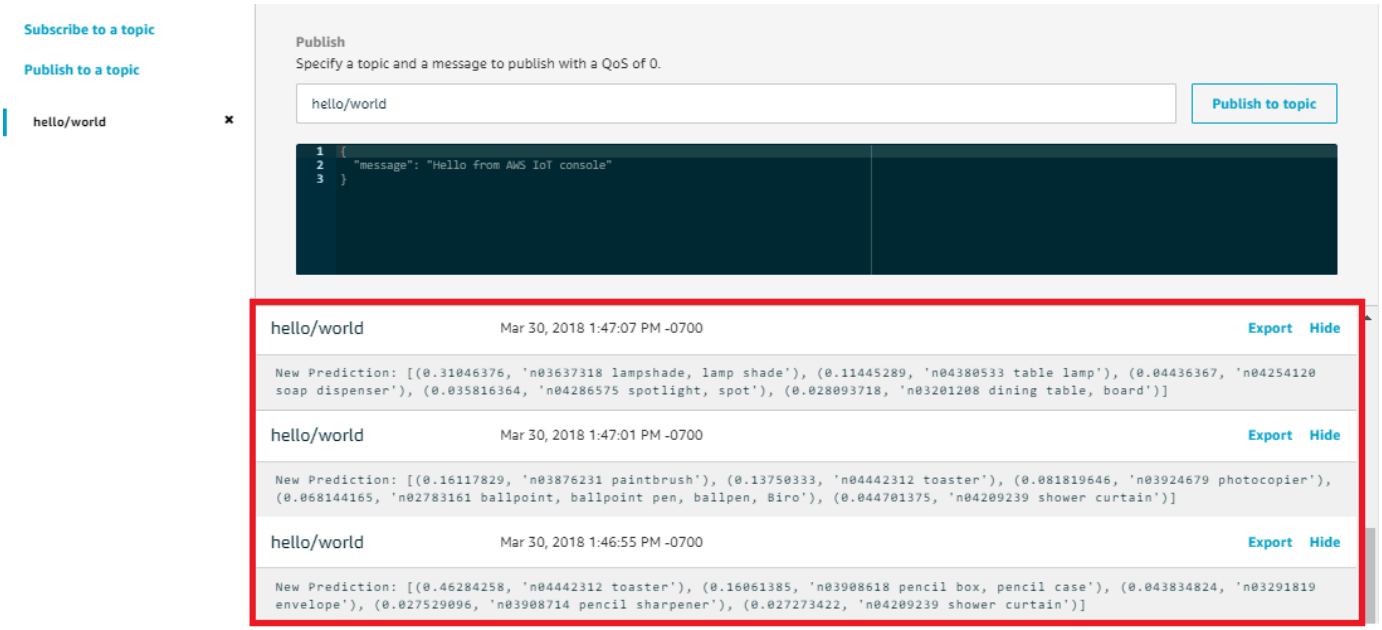
Langkah 9: Tes aplikasi inferensi

Sekarang Anda dapat memverifikasi apakah deployment dikonfigurasi dengan benar. Untuk menguji, Anda berlangganan topik `hello/world` dan melihat hasil prediksi yang diterbitkan oleh fungsi Lambda.

> **Note**
>
> Jika monitor terpasang ke Raspberry Pi, umpan kamera aktif ditampilkan pada jendela pratinjau.

1. DiAWS IoTkonsol, di bawahTEST, PilihKlien uji MQTT.
2. Untuk Berlangganan, gunakan nilai-nilai berikut:
   - Untuk topik berlangganan, gunakan hello/world.
   - Di bawahKonfigurasi tambahan, untukTampilan muatan MQTT, PilihTampilkan muatan sebagai string.
3. Pilih Langganan.

   Jika tes berhasil, pesan dari fungsi Lambda muncul di bagian bawah halaman. Setiap pesan berisi lima hasil prediksi dari citra, menggunakan format: probabilitas, ID kelas diprediksi, dan nama kelas yang sesuai.



Penyelesaian Masalah AWS IoT Greengrass inferensi ML

Jika tes tidak berhasil, Anda dapat mencoba langkah-langkah penyelesaian masalah berikut. Jalankan perintah di terminal Raspberry Pi Anda.

Periksa catatan error

1. Beralih ke pengguna root dan arahkan ke direktori `log` tersebut. Akses ke catatan AWS IoT Greengrass memerlukan izin root.

   ```
   sudo su
   cd /greengrass/ggc/var/log
   ```

2. Di direktori `system` tersebut, periksa `runtime.log` atau `python_runtime.log`.

   Di direktori `user/`*`region`*`/`*`account-id`* tersebut, periksa `greengrassObjectClassification.log`.

   Untuk informasi selengkapnya, lihat the section called “Pemecahan masalah dengan catatan”.

Membongkarkesalahanruntime.log

Jika `runtime.log` berisi error yang mirip dengan berikut ini, pastikan bahwa paket model sumber `tar.gz` Anda memiliki direktori induk.

```
Greengrass deployment error: unable to download the artifact model-arn: Error while
 processing.
Error while unpacking the file from /tmp/greengrass/artifacts/model-arn/path to /
greengrass/ggc/deployment/path/model-arn,
error: open /greengrass/ggc/deployment/path/model-arn/squeezenet/
squeezenet_v1.1-0000.params: no such file or directory
```

Jika paket Anda tidak memiliki direktori induk yang berisi file model, gunakan perintah berikut untuk repackage model tersebut:

```
tar -zcvf model.tar.gz ./model
```

Misalnya:

```
#$ tar -zcvf test.tar.gz ./test
```

```
./test
./test/some.file
./test/some.file2
./test/some.file3
```

> **Note**
>
> Jangan sertakan trailing karakter `/*` dalam perintah ini.

Verifikasi bahwa fungsi Lambda berhasil di-deploy

1. Daftar isi dari Lambda di-deploy di direktori `/lambda` tersebut. Mengganti nilai placeholder sebelum Anda menjalankan perintah.

   ```
   cd /greengrass/ggc/deployment/lambda/
   arn:aws:lambda:region:account:function:function-name:function-version
   ls -la
   ```

2. Verifikasi bahwa direktori berisi konten yang sama sebagai paket deployment `greengrassObjectClassification.zip` yang Anda unggah di Langkah 4: Buat dan publikasikan fungsi Lambda.

   Pastikan bahwa file `.py` dan dependensi berada di root direktori.

Verifikasi bahwa model inferensi berhasil di-deploy

1. Cari nomor identifikasi proses (PID) dari proses waktu aktif Lambda:

   ```
   ps aux | grep 'lambda-function-name*'
   ```

   Pada output, PID muncul di kolom kedua baris untuk proses waktu aktif Lambda.

2. Masukkan namespace waktu aktif Lambda. Pastikan untuk mengganti nilai placeholder *`pid`* sebelum Anda menjalankan perintah.

> **Note**
>
> Direktori ini dan isinya berada di namespace waktu aktif Lambda, sehingga mereka tidak terlihat dalam namespace Linux biasa.

```
sudo nsenter -t pid -m /bin/bash
```

3. Daftar isi dari direktori lokal yang Anda tentukan untuk sumber daya ML.

```
cd /greengrass-machine-learning/mxnet/squeezenet/
ls -ls
```

Anda akan melihat file berikut ini:

```
32 -rw-r--r-- 1 ggc_user ggc_group   31675 Nov 18 15:19 synset.txt
32 -rw-r--r-- 1 ggc_user ggc_group   28707 Nov 18 15:19 squeezenet_v1.1-symbol.json
4832 -rw-r--r-- 1 ggc_user ggc_group 4945062 Nov 18 15:19
 squeezenet_v1.1-0000.params
```

Langkah selanjutnya

Selanjutnya, jelajahi aplikasi inferensi lainnya. AWS IoT Greengrass menyediakan fungsi Lambda lain yang dapat Anda gunakan untuk mencoba inferensi lokal. Anda dapat menemukan contoh paket di folder pustaka precompiled yang Anda unduh di the section called “Instal kerangka kerja MXNet”.

Mengonfigurasi Intel Atom

Untuk menjalankan tutorial ini pada perangkat Intel Atom, Anda harus memberikan citra sumber, Mengonfigurasi fungsi Lambda, dan menambahkan sumber daya perangkat lokal lain. Untuk menggunakan GPU sebagai inferensi, pastikan perangkat lunak berikut diinstal pada perangkat Anda:

- OpenCL versi 1.0 atau yang lebih baru
- Python 3.7 dan pip

> **Note**
>
> Jika perangkat Anda termasuk prebuilt dengan Python 3.6, Anda dapat membuat symlink ke Python 3.7 sebagai gantinya. Untuk informasi selengkapnya, lihat Step 2.

- NumPy
- OpenCV pada Wheels

1. Unduh citra PNG atau JPG statis untuk fungsi Lambda yang akan digunakan untuk klasifikasi citra. Contoh tersebut bekerja paling baik dengan file citra kecil.

   Simpan file citra Anda di direktori yang memuat file `greengrassObjectClassification.py` (atau dalam subdirektori dari direktori ini). Hal ini ada dalam paket deployment fungsi Lambda yang Anda unggah di the section called “Buat dan publikasikan fungsi Lambda”.

   > **Note**
   >
   > Jika Anda menggunakan AWS DeepLens, Anda dapat menggunakan kamera onboard atau memasang kamera Anda sendiri untuk melakukan inferensi pada citra yang ditangkap, bukan citra statis. Namun, kami sangat merekomendasikan Anda mulai dengan citra statis terlebih dahulu.
   > Jika Anda menggunakan kamera, pastikan bahwa paket `awscam` APT diinstal dan diperbarui. Untuk informasi lebih lanjut, lihat Perbarui perangkat AWS DeepLens Anda pada AWS DeepLens Panduan Developer.

2. Jika anda tidak menggunakan Python 3.7, pastikan untuk membuat symlink dari Python 3.x ke Python 3.7. Ini mengonfigurasi perangkat anda untuk menggunakan Python 3 dengan AWS IoT Greengrass. Jalankan perintah berikut untuk menemukan instalasi Python Anda:

   ```
   which python3
   ```

   Jalankan perintah berikut untuk membuat symlink:

   ```
   sudo ln -s path-to-python-3.x/python3.x path-to-python-3.7/python3.7
   ```

   Reboot perangkat.

3. Edit konfigurasi fungsi Lambda. Ikuti prosedur di the section called “Menambahkan fungsi Lambda ke grup”.

> **Note**
>
> Kami merekomendasikan Anda menjalankan fungsi Lambda Anda tanpa kontainerisasi kecuali kasus bisnis Anda memerlukannya. Hal ini membantu mengaktifkan akses ke perangkat GPU dan kamera Anda tanpa mengonfigurasi sumber daya perangkat. Jika Anda menjalankan tanpa kontainerisasi, Anda juga harus memberikan akses root ke Fungsi Lambda AWS IoT Greengrass Anda.

   a. Untuk berjalan tanpa kontainerisasi:

   - UntukPengguna dan grup sistem, Pilih**`Another user ID/group ID`**. UntukID pengguna SistemENTER**`0`**. UntukID Grup SistemENTER**`0`**.

     Hal ini memungkinkan fungsi Lambda Anda untuk berjalan sebagai root. Untuk informasi lebih lanjut untuk menjalankan sebagai root, lihat the section called “Mengatur identitas akses default untuk fungsi Lambda dalam grup”.

     > **Tip**
     >
     > Anda juga harus memperbarui file `config.json` untuk memberikan akses root ke fungsi Lambda Anda. Untuk prosedur ini, lihat the section called “Menjalankan fungsi Lambda sebagai root”.

   - UntukFungsi Lambda kontainerisasi, PilihTanpa kontainer.

     Untuk informasi lebih lanjut untuk berjalan tanpa konainerisasi, lihat the section called “Pertimbangan ketika memilih fungsi Lambda kontainerisasi”.

   - Perbarui nilai Timeout menjadi 5 detik. Hal ini memastikan bahwa permintaan tidak timeout terlalu cepat. Dibutuhkan beberapa menit setelah pengaturan untuk menjalankan inferensi.
   - Di bawahDipinilih, PilihBenar.
   - Di bawahParameter tambahan, untukBaca akses ke direktori /sys, PilihDiaktifkan.

- Untuk Siklus hidup Lambda, pilih Jadikan fungsi ini berumur panjang dan biarkan berjalan tanpa batas.

b. Untuk menjalankan dalam mode kontainer sebagai gantinya:

> **Note**
>
> Kami tidak merekomendasikan untuk dijalankan dalam mode kontainer kecuali jika kasus bisnis Anda memerlukannya.

- Perbarui nilai Timeout menjadi 5 detik. Hal ini memastikan bahwa permintaan tidak timeout terlalu cepat. Dibutuhkan beberapa menit setelah pengaturan untuk menjalankan inferensi.
- UntukDipinilih, PilihBenar.
- Di bawahParameter tambahan, untukBaca akses ke direktori /sys, PilihDiaktifkan.

4. Jika berjalan dalam mode kontainerisasi, tambahkan sumber daya perangkat lokal yang diperlukan untuk memberikan akses ke GPU perangkat Anda.

> **Note**
>
> Jika Anda menjalankan dalam mode nonkontainerisasi, AWS IoT Greengrass dapat mengakses GPU perangkat Anda tanpa mengonfigurasi sumber daya perangkat.

a. Pada halaman konfigurasi grup, pilihSumber dayaTab.

b. PilihMenambahkan sumber daya lokal.

c. Mendefinisikan sumber daya:

- Untuk Nama sumber daya, masukkan **`renderD128`**.
- UntukJenis sumber daya, PilihPerangkat lokal.
- Untuk Jalur perangkat, masukkan **`/dev/dri/renderD128`**.
- UntukPemilik grup sistem dan izin akses file, PilihSecara otomatis menambahkan izin sistem file dari grup sistem yang memiliki sumber daya.
- Untuk Afiliasi fungsi Lambda, berikan Akses baca dan tulis pada fungsi Lambda Anda.

Mengonfigurasi NVIDIA Jetson TX2

Untuk menjalankan tutorial ini pada NVIDIA Jetson TX2, berikan citra sumber dan konfigurasi fungsi Lambda. Jika menggunakan GPU, Anda juga harus menambahkan sumber daya perangkat lokal.

1. Pastikan perangkat Jetson Anda dikonfigurasi sehingga Anda dapat menginstal perangkat lunak AWS IoT Greengrass core. Untuk informasi lebih lanjut tentang konfigurasi perangkat Anda, lihat the section called “Mengatur perangkat lain”.
2. Buka dokumentasi MXNet, pergi ke Menginstal MXNet pada Jetson, lalu ikuti petunjuk untuk menginstal MXNet pada perangkat Jetson.

   > **Note**
   >
   > Jika Anda ingin membangun MXNet dari sumber, ikuti petunjuk untuk membuat perpustakaan berbagi. Edit pengaturan berikut dalam file `config.mk` Anda agar bekerja dengan perangkat Jetson TX2:
   >
   > - Tambahkan `-gencode arch=compute-62, code=sm_62` ke pengaturan `CUDA_ARCH` tersebut.
   > - Hidupkan CUDA.
   >
   > ```
   > USE_CUDA = 1
   > ```

3. Unduh citra PNG atau JPG statis untuk fungsi Lambda yang akan digunakan untuk klasifikasi citra. Aplikasi ini bekerja paling baik dengan file citra kecil. Atau, Anda dapat menyetel kamera melalui forum Jetson untuk menangkap citra sumber.

   Simpan file citra Anda di direktori yang memuat file `greengrassObjectClassification.py` tersebut. Anda juga dapat menyimpannya dalam subdirektori di direktori ini. Direktori ini ada dalam paket deployment fungsi Lambda yang Anda unggah di the section called “Buat dan publikasikan fungsi Lambda”.
4. Buat symlink dari Python 3.7 ke Python 3.6 agar menggunakan Python 3 dengan AWS IoT Greengrass. Jalankan perintah berikut untuk menemukan instalasi Python Anda:

   ```
   which python3
   ```

   Jalankan perintah berikut untuk membuat symlink:

```
sudo ln -s path-to-python-3.6/python3.6 path-to-python-3.7/python3.7
```

Reboot perangkat.

5. Pastikan akun sistem `ggc_user` dapat menggunakan kerangka kerja MXNet:

```
“sudo -u ggc_user bash -c 'python3 -c "import mxnet"'
```

6. Edit konfigurasi fungsi Lambda. Ikuti prosedur di the section called “Menambahkan fungsi Lambda ke grup”.

> **Note**
>
> Kami merekomendasikan Anda menjalankan fungsi Lambda Anda tanpa kontainerisasi kecuali kasus bisnis Anda memerlukannya. Hal ini membantu mengaktifkan akses ke perangkat GPU dan kamera Anda tanpa mengonfigurasi sumber daya perangkat. Jika Anda menjalankan tanpa kontainerisasi, Anda juga harus memberikan akses root ke Fungsi Lambda AWS IoT Greengrass Anda.

a. Untuk berjalan tanpa kontainerisasi:

- UntukPengguna dan grup sistem, Pilih**Another user ID/group ID**. UntukID pengguna SistemENTER**0**. UntukID Grup SistemENTER**0**.

  Hal ini memungkinkan fungsi Lambda Anda untuk berjalan sebagai root. Untuk informasi lebih lanjut untuk menjalankan sebagai root, lihat the section called “Mengatur identitas akses default untuk fungsi Lambda dalam grup”.

  > **Tip**
  >
  > Anda juga harus memperbarui file `config.json` untuk memberikan akses root ke fungsi Lambda Anda. Untuk prosedur ini, lihat the section called “Menjalankan fungsi Lambda sebagai root”.

- UntukFungsi Lambda kontainerisasi, PilihTanpa kontainer.

  Untuk informasi lebih lanjut untuk berjalan tanpa konainerisasi, lihat the section called “Pertimbangan ketika memilih fungsi Lambda kontainerisasi”.

- Di bawahParameter tambahan, untukBaca akses ke direktori /sys, PilihDiaktifkan.
- Di bawah Variabel lingkungan, tambahkan pasangan nilai kunci berikut untuk fungsi Lambda Anda. Ini mengonfigurasi AWS IoT Greengrass untuk menggunakan kerangka kerja MXNet.

Kunci	Nilai
PATH	/usr/local/cuda/bin: $PATH
MXNET_HOME	$ rumah/mxnet/
PYTHONPATH	$mxnet_HOME/Python: $PYTHONPATH
CUDA_HOME	/usr/local/cuda
LD_LIBRARY_PATH	$ LD_LIBRARY_PATH: $ {CUDA_HOM E} /lib64

b. Untuk menjalankan dalam mode kontainer sebagai gantinya:

> **Note**
>
> Kami tidak merekomendasikan untuk dijalankan dalam mode kontainer kecuali jika kasus bisnis Anda memerlukannya.

- Meningkatkan nilai Batas memori tersebut. Gunakan 500 MB untuk CPU, atau setidaknya 2000 MB untuk GPU.
- Di bawahParameter tambahan, untukBaca akses ke direktori /sys, PilihDiaktifkan.
- Di bawah Variabel lingkungan, tambahkan pasangan nilai kunci berikut untuk fungsi Lambda Anda. Ini mengonfigurasi AWS IoT Greengrass untuk menggunakan kerangka kerja MXNet.

Kunci	Nilai
PATH	/usr/local/cuda/bin: $PATH

Kunci	Nilai
MXNET_HOME	$ rumah/mxnet/
PYTHONPATH	$mxnet_HOME/Python: $PYTHONPATH
CUDA_HOME	/usr/local/cuda
LD_LIBRARY_PATH	$ LD_LIBRARY_PATH: $ {CUDA_HOM E} /lib64

7. Jika berjalan dalam mode kontainerisasi, tambahkan sumber daya perangkat lokal berikut untuk memberikan akses ke GPU perangkat Anda. Ikuti prosedur di the section called “Menambahkan sumber daya ke grup”.

   > **Note**
   >
   > Jika Anda menjalankan dalam mode nonkontainerisasi, AWS IoT Greengrass dapat mengakses GPU perangkat Anda tanpa mengonfigurasi sumber daya perangkat.

   Untuk setiap sumber daya:

   - Untuk Jenis sumber daya, pilih Perangkat.
   - UntukPemilik grup sistem dan izin akses file, PilihSecara otomatis menambahkan izin sistem file dari grup sistem yang memiliki sumber daya.

Nama	Jalur perangkat
nvhost-ctrl	/dev/nvhost-ctrl
nvhost-gpu	/dev/nvhost-gpu
nvhost-ctrl-gpu	/dev/dev/dev/nvhost-ctrl-gpu
nvhost-dbg-gpu	/dev/dev/dev/nvhost-dbg-gpu

Nama	Jalur perangkat
nvhost-prof-gpu	/dev/dev/dev/nvhost-prof-gpu
nvmap	/dev/nvmap
nvhost-vic	/dev/nvhost-vic
tegra_dc_ctrl	/dev/tegra_dc_ctrl

8. Jika berjalan dalam mode kontainerisasi, tambahkan sumber daya volume lokal berikut untuk memberikan akses ke kamera perangkat Anda. Ikuti prosedur di the section called "Menambahkan sumber daya ke grup".

> **Note**
>
> Jika Anda menjalankan dalam mode nonkontainerisasi, AWS IoT Greengrass dapat mengakses kamera perangkat Anda tanpa mengonfigurasi sumber daya volume.

   - Untuk Jenis sumber daya, pilih Volume.
   - UntukPemilik grup sistem dan izin akses file, PilihSecara otomatis menambahkan izin sistem file dari grup sistem yang memiliki sumber daya.

Nama	Jalur sumber	Jalur tujuan
shm	/dev/shm	/dev/shm
tmp	/tmp	/tmp

Cara mengonfigurasi kesimpulan machine learning yang dioptimalkan menggunakan AWS Management Console

Untuk mengikuti langkah-langkah di tutorial ini, Anda harus menggunakan AWS IoT Greengrass Core v1.10 atau yang lebih baru.

Anda dapat menggunakan SageMaker Kompilator deep learning Neo untuk mengoptimalkan efisiensi prediksi model inferensi machine learning asli di Tensorflow, Apache MXNet, PyTorchFramework, ONNX, dan XGBoost untuk footprint yang lebih kecil dan kinerja yang lebih cepat. Anda kemudian dapat mengunduh model yang dioptimalkan dan menginstal SageMaker Neo deep learning deep learning dan terapkan keAWS IoT Greengrassperangkat untuk inferensi lebih cepat.

Tutorial ini menjelaskan cara menggunakan AWS Management Console untuk mengonfigurasi grup Greengrass untuk menjalankan contoh inferensi Lambda yang mengenali citra dari kamera secara lokal, tanpa mengirim data ke cloud. Contoh kesimpulan mengakses modul kamera pada Raspberry Pi. Dalam tutorial ini, Anda mengunduh model prepackaged yang dilatih oleh Resnet-50 dan dioptimalkan di kompilator Neo deep learning. Anda kemudian menggunakan model untuk melakukan klasifikasi citra lokal pada perangkat AWS IoT Greengrass Anda.

Tutorial ini berisi langkah-langkah tingkat tinggi berikut:

1. Mengonfigurasi Raspberry Pi
2. PasangNeo deep learning deep learning
3. Buat fungsi inferensi Lambda
4. Tambahkan fungsi Lambda ke grup
5. Tambahkan sumber daya model NEO-dioptimalkan ke grup
6. Tambahkan sumber daya perangkat kamera ke grup
7. Tambahkan langganan ke grup
8. Men-deploy grup
9. Uji contoh

Prasyarat

Untuk menyelesaikan tutorial ini, Anda memerlukan:

- Raspberry Pi 4 Model B, atau Raspberry Pi 3 Model B/B+, diatur dan dikonfigurasi untuk digunakan dengan AWS IoT Greengrass. Untuk mengatur Raspberry Pi Anda dengan AWS IoT Greengrass, jalankan Pengaturan Perangkat Greengrass tersebut, atau pastikan bahwa Anda telah menyelesaikan Modul 1 dan Modul 2 dari Mulai menggunakan AWS IoT Greengrass.

> **Note**
>
> Raspberry Pi mungkin memerlukan Catu daya 2.5A untuk menjalankan kerangka kerja deep learning yang biasanya digunakan untuk klasifikasi citra. Catu daya dengan peringkat lebih rendah dapat menyebabkan perangkat melakukan reboot.

- Modul Kamera Raspberry Pi V2 - 8 megapiksel, 1080p. Untuk mempelajari cara mengatur kamera, lihat Menghubungkan kamera dalam dokumentasi Raspberry Pi.
- Sebuah grup Greengrass dan Greengrass core. Untuk mempelajari cara membuat grup Greengrass atau core, lihat Mulai menggunakan AWS IoT Greengrass.

> **Note**
>
> Tutorial ini menggunakan Raspberry Pi, AWS IoT Greengrass mendukung platform lain, seperti Intel Atom dan NVIDIA Jetson TX2. Jika menggunakan contoh Intel Atom, Anda mungkin butuh menginstal Python 3.6 bukannya Python 3.7. Untuk informasi tentang cara mengonfigurasi perangkat sehingga Anda dapat menginstal AWS IoT Greengrass perangkat lunak Core, lihat the section called “Mengatur perangkat lain”.
> Untuk platform pihak ketiga di mana AWS IoT Greengrass tidak mendukung, Anda harus menjalankan fungsi Lambda Anda dalam mode nonkontainerisasi. Untuk menjalankan dalam mode nonkontainerisasi, Anda harus menjalankan fungsi Lambda Anda sebagai root. Untuk informasi lebih lanjut, lihat the section called “Pertimbangan ketika memilih fungsi Lambda kontainerisasi” dan the section called “Mengatur identitas akses default untuk fungsi Lambda dalam grup”.

Langkah 1: Mengonfigurasi Raspberry Pi

Pada langkah ini, instal pembaruan untuk sistem operasi Raspbian, instal perangkat lunak modul kamera dan Python dependensi, dan mengaktifkan antarmuka kamera.

Jalankan perintah berikut di terminal Raspberry Pi Anda.

1. Menginstal pembaruan untuk Raspbian.

```
sudo apt-get update
```

```
sudo apt-get dist-upgrade
```

2. Instal antarmuka `picamera` untuk modul kamera dan pustaka Python lain yang diperlukan untuk tutorial ini.

   ```
   sudo apt-get install -y python3-dev python3-setuptools python3-pip python3-picamera
   ```

   Memvalidasi instalasi:

   - Pastikan bahwa instalasi Python 3.7 anda termasuk pip.

     ```
     python3 -m pip
     ```

     Jika pip tidak terpasang, unduh dari Situs web pip tersebut, kemudian jalankan perintah berikut.

     ```
     python3 get-pip.py
     ```

   - Pastikan bahwa versi Python Anda adalah 3.7 atau lebih tinggi.

     ```
     python3 --version
     ```

     Jika output mencantumkan versi sebelumnya, jalankan perintah berikut.

     ```
     sudo apt-get install -y python3.7-dev
     ```

   - Pastikan bahwa Setuptools dan Picamera berhasil diinstal.

     ```
     sudo -u ggc_user bash -c 'python3 -c "import setuptools"'
     sudo -u ggc_user bash -c 'python3 -c "import picamera"'
     ```

     Jika output tidak mengandung error, validasi berhasil.

   > **Note**
   >
   > Jika Python yang dapat dieksekusi diinstal pada perangkat anda adalah `python3.7`, gunakan `python3.7` daripada `python3` untuk perintah di tutorial ini. Pastikan bahwa

instalasi pip Anda memetakan versi `python3.7` atau `python3` yang tepat untuk menghindari kesalahan dependensi.

3. Reboot Raspberry Pi.

```
sudo reboot
```

4. Buka alat konfigurasi Raspberry Pi.

```
sudo raspi-config
```

5. Gunakan tombol panah untuk membuka Opsi Antarmuka dan mengaktifkan antarmuka kamera. Jika diminta, izinkan perangkat melakukan reboot.
6. Gunakan perintah berikut untuk menguji pengaturan kamera.

```
raspistill -v -o test.jpg
```

Hal ini akan membuka jendela pratinjau pada Raspberry Pi, menyimpan gambar bernama `test.jpg` ke direktori Anda saat ini, dan menampilkan informasi tentang kamera di terminal Raspberry Pi.

Langkah 2: Instal Amazon SageMaker Waktu aktif deep learning

Dalam langkah ini, menginstal waktu aktif deep learning (DLR) Neo pada Anda Raspberry Pi.

> **Note**
>
> Kami sarankan menginstal versi 1.1.0 untuk tutorial ini.

1. Masuk ke Raspberry Pi Anda secara jarak jauh.

```
ssh pi@your-device-ip-address
```

2. Buka dokumentasi DLR, buka Menginstal DLR, dan menemukan URL wheel untuk perangkat Raspberry Pi. Kemudian, ikuti petunjuk untuk menginstal DLR di perangkat Anda. Sebagai contoh, Anda dapat menggunakan pip:

```
pip3 install rasp3b-wheel-url
```

3. Setelah Anda menginstal DLR, validasikan konfigurasi berikut:

   - Pastikan `ggc_user` akun sistem dapat menggunakan perpustakaan DLR.

     ```
     sudo -u ggc_user bash -c 'python3 -c "import dlr"'
     ```

   - Pastikan NumPy dipasang.

     ```
     sudo -u ggc_user bash -c 'python3 -c "import numpy"'
     ```

Langkah 3: Buat fungsi inferensi Lambda

Pada langkah ini, buat paket deployment fungsi Lambda dan fungsi Lambda. Kemudian, terbitkan versi fungsi dan membuat alias.

1. Pada komputer Anda, unduh sampel DLR untuk Raspberry Pi dari the section called “Sampel machine learning”.
2. Unzip yang diunduh `dlr-py3-armv7l.tar.gz` File.

   ```
   cd path-to-downloaded-sample
   tar -xvzf dlr-py3-armv7l.tar.gz
   ```

   Direktori `examples` dalam paket contoh yang diekstrak berisi kode fungsi dan dependensi.

   - `inference.py` adalah kode inference yang digunakan dalam tutorial ini. Anda dapat menggunakan kode ini sebagai templat untuk membuat fungsi inferensi Anda sendiri.
   - `greengrasssdk` adalah versi 1.5.0 dari AWS IoT Greengrass Core SDK for Python.

   > **Note**
   >
   > Jika versi baru tersedia, Anda dapat mengunduh dan membarui versi SDK dalam paket deployment Anda. Untuk informasi selengkapnya, lihatAWS IoT GreengrassCore SDK for Pythondi atas GitHub.

3. Kompres isi `examples` ke dalam sebuah file bernama `optimizedImageClassification.zip`. Ini paket deployment Anda.

```
cd path-to-downloaded-sample/dlr-py3-armv7l/examples
zip -r optimizedImageClassification.zip .
```

Paket deployment berisi dependensi dan kode fungsi Lambda Anda. Ini termasuk kode yang memanggil Neo deep learning runtime Python API untuk melakukan inferensi dengan model kompilator Neo deep learning.

> **Note**
>
> Pastikan `.py` file dan dependensi berada di root direktori.

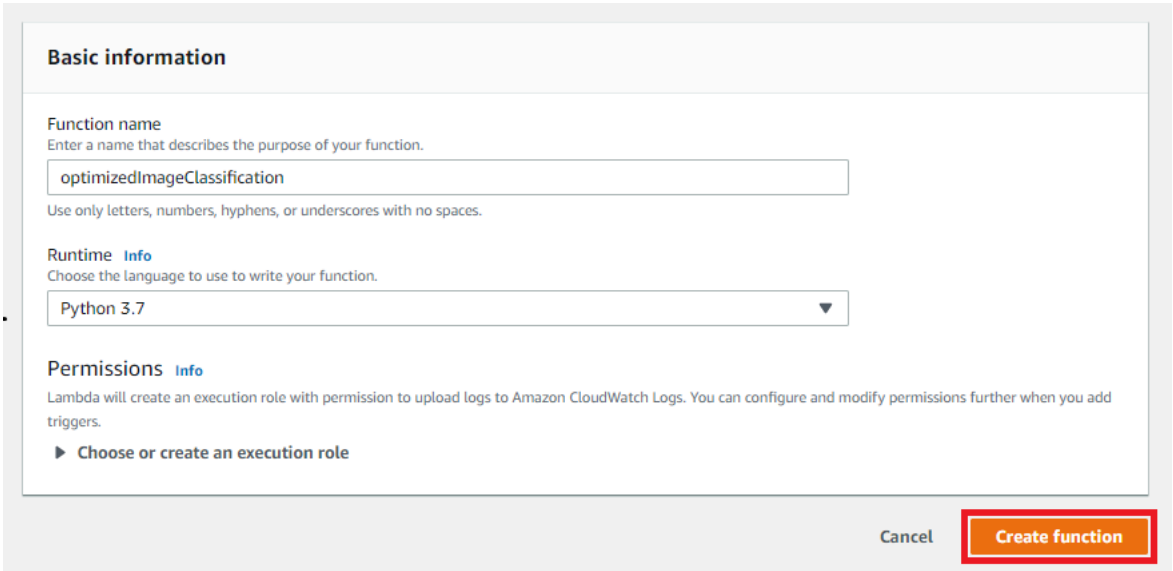
4. Sekarang, tambahkan fungsi Lambda ke grup Greengrass Anda.

Dari halaman konsol Lambda, pilihFungsidan pilihlahBuat fungsi.

5. Pilih Tulis dari awal dan gunakan nilaiinilai berikut untuk membuat fungsi Anda:

- Untuk Nama fungsi, masukkan **`optimizedImageClassification`**.
- Untuk Waktu pengoperasian, pilih Python 3.7.

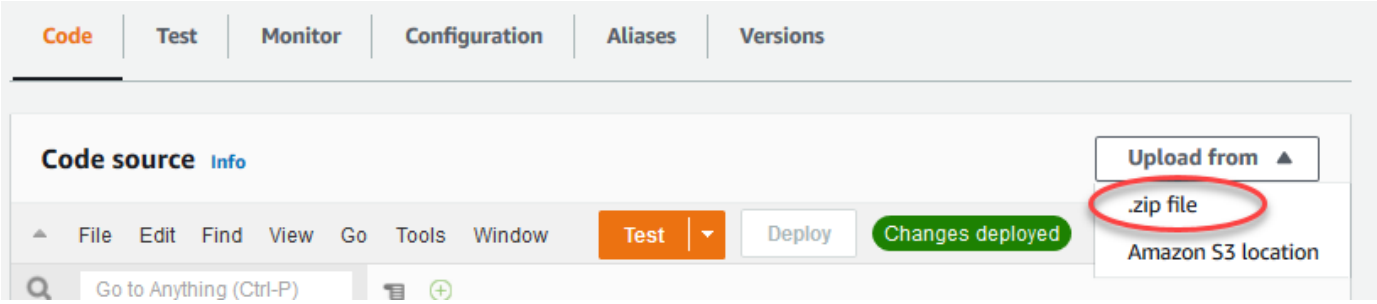
Untuk Izin, pertahankan pengaturan default. Hal ini menciptakan peran eksekusi yang memberikan izin Lambda basic. Peran ini tidak digunakan oleh AWS IoT Greengrass.



6. Pilih Buat fungsi.

Sekarang, unggah paket deployment fungsi Lambda Anda dan daftarkan handler.

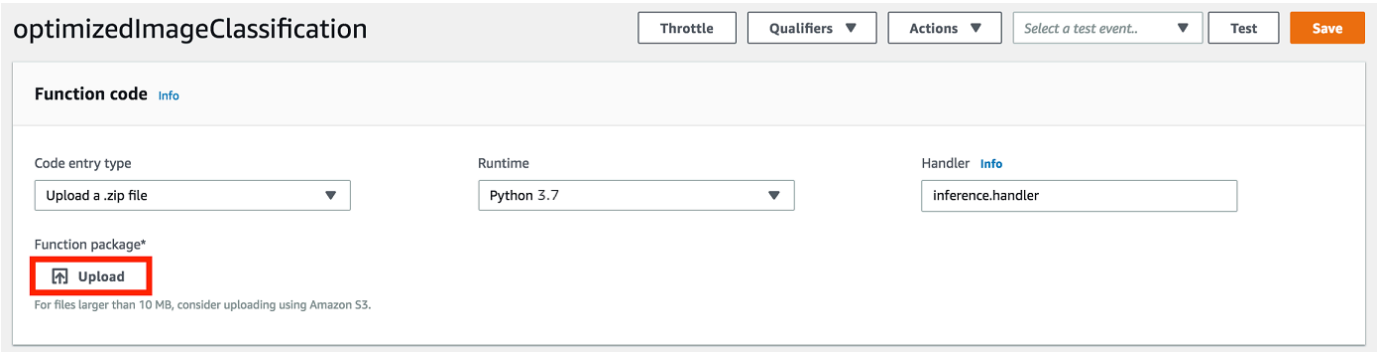
1. Pada tab Kode ini, di bawah Sumber kode, pilih Unggah dari. Dari dropdown, pilih file .zip.



2. Pilih paket deployment `optimizedImageClassification.zip` Anda, dan kemudian pilih Simpan.
3. Pada tab Kode fungsi, di bawah Pengaturan waktu aktif, pilih Edit, dan kemudian masukkan nilai-nilai berikut.

   - Untuk waktu aktif, pilih Python 3.7.
   - Untuk Handler, masukkan **`inference.handler`**.

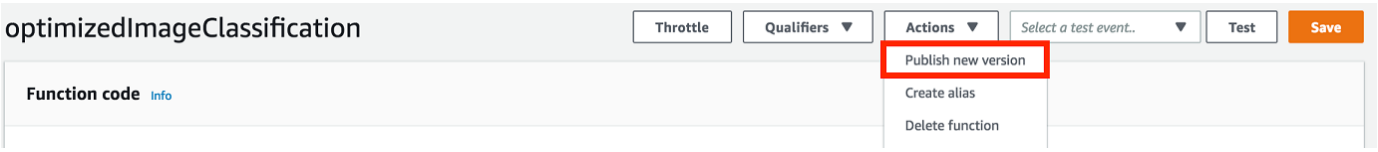
   Pilih Save (Simpan).



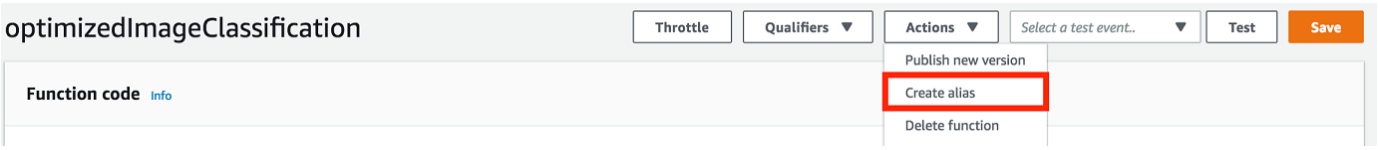
Selanjutnya, terbitkan versi pertama fungsi Lambda Anda. Kemudian, buat alias untuk versi.

> **Note**
>
> Grup Greengrass dapat mereferensi fungsi Lambda dengan alias (direkomendasikan) atau dengan versi. Menggunakan alias membuatnya lebih mudah untuk mengelola pembaruan kode karena Anda tidak perlu mengubah tabel langganan atau definisi grup ketika kode fungsi diperbarui. Sebaliknya, Anda hanya mengarahkan alias ke versi fungsi baru.

1. Dari menu Tindakan tersebut, pilih Publikasikan versi baru.



2. Untuk Versi Deskripsi, masukkan **`First version`**, lalu pilih Publikasikan.
3. PadaoptimizedImageClassification: 1halaman konfigurasi, dariTindakanmenu, pilihMembuat alias.



4. Pada halaman Buat alias baru tersebut, gunakan nilai-nilai berikut:

   - Untuk Nama, masukkan **`mlTestOpt`**.
   - Untuk UID, masukkan **`1`**.

   > **Note**
   >
   > AWS IoT Greengrass tidak support alias Lambda untuk versi $TERBARU ini.

5. Pilih Create (Buat).

   Sekarang, tambahkan fungsi Lambda ke grup Greengrass Anda.

Langkah 4: Tambahkan fungsi Lambda ke grup Greengrass

Pada langkah ini, tambahkan fungsi Lambda ke grup, dan kemudian mengonfigurasi siklus hidupnya.

Pertama, tambahkan fungsi Lambda ke grup Greengrass Anda.

1. DiAWS IoTpanel navigasi konsol, di bawahKelola, PerluasPerangkat Greengrass, dan kemudian pilihGrup (V1).
2. Pada halaman konfigurasi grup, pilihFungsi Lambdatab, dan pilihTambahkan.
3. PilihFungsi Lambdadan pilihlahoptimizedImageClassification.
4. PadaVersi fungsi Lambda, pilih alias ke versi yang Anda terbitkan.

Selanjutnya, konfigurasikan siklus hidup fungsi Lambda.

1. DiKonfigurasi fungsi Lambdabagian, buat pembaruan berikut.

   > **Note**
   >
   > Kami merekomendasikan Anda menjalankan fungsi Lambda Anda tanpa kontainerisasi kecuali kasus bisnis Anda memerlukannya. Hal ini membantu mengaktifkan akses ke perangkat GPU dan kamera Anda tanpa mengonfigurasi sumber daya perangkat. Jika Anda menjalankan tanpa kontainerisasi, Anda juga harus memberikan akses root ke Fungsi Lambda AWS IoT Greengrass Anda.

   a. Untuk berjalan tanpa kontainerisasi:

      - UntukPengguna dan grup sistem, choose**`Another user ID/group ID`**. UntukID Pengguna SistemENTER**`0`**. UntukID Grup SistemENTER**`0`**.

        Hal ini memungkinkan fungsi Lambda Anda untuk berjalan sebagai root. Untuk informasi lebih lanjut untuk menjalankan sebagai root, lihat the section called “Mengatur identitas akses default untuk fungsi Lambda dalam grup”.

        > **Tip**
        >
        > Anda juga harus memperbarui file `config.json` untuk memberikan akses root ke fungsi Lambda Anda. Untuk prosedur ini, lihat the section called “Menjalankan fungsi Lambda sebagai root”.

      - UntukFungsi Lambda kontainerisasi, chooseTanpa kontainer.

Untuk informasi lebih lanjut untuk berjalan tanpa konainerisasi, lihat the section called “Pertimbangan ketika memilih fungsi Lambda kontainerisasi”.

- Untuk Timeout, masukkan **`10 seconds`**.
- UntukDipinned, chooseBenar.

  Untuk informasi selengkapnya, lihat the section called “Konfigurasi siklus hidup”.
- Di bawahParameter tambahan, untukBaca akses ke direktori /sys, chooseDiaktifkan.

b. Untuk menjalankan dalam mode kontainer sebagai gantinya:

> **Note**
>
> Kami tidak merekomendasikan untuk dijalankan dalam mode kontainer kecuali jika kasus bisnis Anda memerlukannya.

- UntukPengguna dan grup sistem, chooseGunakan default grup.
- UntukFungsi Lambda kontainerisasi, chooseGunakan default grup.
- Untuk Batas memori, masukkan **`1024 MB`**.
- Untuk Timeout, masukkan **`10 seconds`**.
- UntukDipinned, chooseBenar.

  Untuk informasi selengkapnya, lihat the section called “Konfigurasi siklus hidup”.
- Di bawahParameter tambahan, untukBaca akses ke direktori /sys, chooseDiaktifkan.

2. PilihTambahkan fungsi Lambda.

Langkah 5: Tambahkan SageMaker Sumber daya model Neo-dioptimalkan untuk grup Greengrass

Pada langkah ini, buat sumber daya untuk model inferensi ML dioptimalkan dan unggah ke bucket Amazon S3. Kemudian, tempatkan Amazon S3 yang diunggah di AWS IoT Greengrass konsol dan afiliasi sumber daya yang baru dibuat dengan fungsi Lambda. Hal ini memungkinkan fungsi untuk mengakses sumber dayanya pada perangkat core.

1. Pada komputer Anda, navigasikan ke direktori `resnet50` dalam paket contoh yang Anda unzip di the section called “Buat fungsi inferensi Lambda”.

> **Note**
>
> Jika menggunakan contoh NVIDIA Jetson, Anda harus menggunakan `resnet18` dalam paket contoh sebagai gantinya. Untuk informasi selengkapnya, lihat the section called “Mengkonfigurasi NVIDIA Jetson TX2”.

```
cd path-to-downloaded-sample/dlr-py3-armv7l/models/resnet50
```

Direktori ini berisi artefak model precompiled untuk model klasifikasi citra dilatih dengan Resnet-50.

2. Kompres file di dalam direktori `resnet50` ke dalam sebuah file bernama `resnet50.zip`.

```
zip -r resnet50.zip .
```

3. Pada halaman konfigurasi grup untukAWS IoT Greengrasskelompok, pilihSumber dayaTab. Navigasikan ke Machine Learning bagian dan pilih Tambah sumber daya machine learning. Pada halaman Buat sumber daya machine learning ini, untuk Nama sumber daya, masukkan **`resnet50_model`**.
4. UntukSumber model, chooseGunakan model yang disimpan di S3, seperti model yang dioptimalkan melalui Deep Learning Compiler.
5. Di bawahJENIS S3, chooseMenjelajahi S3.

> **Note**
>
> Saat ini, dioptimalkan SageMaker model disimpan secara otomatis di Amazon S3. Anda dapat menemukan model yang dioptimalkan di bucket Amazon S3 Anda menggunakan opsi ini. Untuk informasi lebih lanjut tentang optimalisasi model di SageMaker, lihatSageMaker Dokumentasi Neo.

6. Pilih Uunggah model.
7. Pada tab konsol Amazon S3, unggah file zip Anda ke bucket Amazon S3. Untuk informasi, lihatBagaimana cara mengunggah file dan folder ke S3 bucket?di dalamPanduan Pengguna Amazon Simple Storage Service.

> **Note**
>
> Nama bucket Anda harus berisi string **`greengrass`**. Pilih nama yang unik (seperti **`greengrass-dlr-bucket-`*`user-id`*`-`*`epoch-time`***). Jangan gunakan periode (`.`) dalam nama bucket.

8. Di AWS IoT Greengrass tab konsol, tempatkan dan pilih bucket Amazon S3. Tempatkan file `resnet50.zip` yang Anda unggah, dan pilih Pilihan. Anda mungkin perlu me-refresh halaman untuk memperbarui daftar bucket dan file yang tersedia.
9. MasukJalur tujuanENTER**`/ml_model`**.

Hal ini adalah tujuan untuk model lokal di Lambda waktu aktif namespace. Ketika Anda men-deploy grup, AWS IoT Greengrass mengambil paket model sumber dan kemudian mengekstraksi isi ke direktori tertentu.

> **Note**
>
> Kami sangat merekomendasikan Anda menggunakan jalur yang tepat yang disediakan untuk jalur lokal Anda. Menggunakan jalur tujuan model lokal yang berbeda dalam langkah ini menyebabkan beberapa perintah pemecahan masalah yang disediakan dalam tutorial ini menjadi tidak akurat. Jika Anda menggunakan jalur yang berbeda, Anda harus mengatur `MODEL_PATH` lingkungan yang menggunakan jalur yang tepat yang Anda berikan di sini. Untuk informasi lebih lanjut tentang variabel lingkungan, lihat AWS Lambda variabel lingkungan.

10. Jika berjalan dalam mode containerized:
    a. Di bawahPemilik grup sistem dan izin akses file, chooseTentukan grup sistem dan izin.
    b. PilihAkses hanya-bacadan kemudian pilihTambahkan sumber daya.

Langkah 6: Tambahkan sumber daya perangkat kamera ke grup Greengrass

Pada langkah ini, buat sumber daya untuk modul kamera dan mengafiliasikannya dengan fungsi Lambda. Hal ini memungkinkan fungsi Lambda untuk mengakses sumber daya pada perangkat core.

> **Note**
>
> Jika Anda berjalan dalam mode nonkontainerisasi, AWS IoT Greengrass dapat mengakses GPU dan kamera perangkat Anda tanpa mengonfigurasi sumber daya perangkat ini.

1. Pada halaman konfigurasi grup, pilihSumber dayaTab.
2. PadaSumber daya lokaltab, pilihTambahkan sumber daya lokal.
3. PadaMenambahkan sumber daya lokalhalaman, gunakan nilai-nilai berikut:

   - Untuk Nama sumber daya, masukkan **`videoCoreSharedMemory`**.
   - Untuk Jenis sumber daya, pilih Perangkat.
   - UntukJalur perangkat lokalENTER**`/dev/vcsm`**.

     Jalur perangkat adalah jalur absolut lokal sumber daya perangkat. Path ini hanya dapat merujuk ke perangkat karakter atau perangkat blok di bawah `/dev`.
   - UntukPemilik grup sistem dan izin akses file, chooseSecara otomatis menambahkan izin sistem file dari grup sistem yang memiliki sumber daya.

     Opsi Izin akses file pemilik grup memungkinkan Anda memberikan izin akses file tambahan untuk proses Lambda. Untuk informasi selengkapnya, lihat Izin akses file pemilik grup.
4. Di bagian bawah halaman, pilihTambahkan sumber daya.
5. DariSumber dayatab, buat sumber daya lokal lain dengan memilihTambahkandan gunakan nilai-nilai berikut:

   - Untuk Nama sumber daya, masukkan **`videoCoreInterface`**.
   - Untuk Jenis sumber daya, pilih Perangkat.
   - UntukJalur perangkat lokalENTER**`/dev/vchiq`**.
   - UntukPemilik grup sistem dan izin akses file, chooseSecara otomatis menambahkan izin sistem file dari grup sistem yang memiliki sumber daya.

6. PilihTambahkan sumber daya.

Langkah 7: Tambahkan langganan ke grup Greengrass

Pada langkah ini, tambahkan langganan ke grup. Langganan ini memungkinkan fungsi Lambda untuk mengirim hasil prediksi ke AWS IoT dengan menerbitkan topik MQTT.

1. Pada halaman konfigurasi grup, pilihLangganantab, dan kemudian pilihTambahkan langganan.
2. PadaBuat langgananhalaman, mengkonfigurasi sumber dan target, sebagai berikut:
   a. MasukJenis sumber, chooseFungsi Lambda, dan kemudian pilihoptimizedImageClassification.
   b. MasukJenis target, chooseLayanan, dan kemudian pilihIoT Cloud.
   c. DiFilter topikENTER**`/resnet-50/predictions`**, dan kemudian pilihBuat langganan.
3. Tambahkan langganan kedua. PilihLangganantab, pilihTambahkan langganan, dan mengonfigurasi sumber dan target, sebagai berikut:
   a. MasukJenis sumber, chooseLayanan, dan kemudian pilihIoT Cloud.
   b. MasukJenis target, chooseFungsi Lambda, dan kemudian pilihoptimizedImageClassification.
   c. DiFilter topikENTER**`/resnet-50/test`**, dan kemudian pilihBuat langganan.

Langkah 8: Men-deploy grup Greengrass

Pada langkah ini, men-deploy versi definisi grup untuk perangkat core Greengrass. Definisi berisi fungsi Lambda, sumber daya, dan konfigurasi langganan yang ditambahkan.

1. Pastikan bahwa core AWS IoT Greengrass sedang berjalan. Jalankan perintah berikut di terminal Raspberry Pi Anda, sesuai kebutuhan.
   a. Untuk memeriksa apakah daemon sedang berjalan:

      ```
      ps aux | grep -E 'greengrass.*daemon'
      ```

      Jika output berisi `root` entri untuk `/greengrass/ggc/packages/`*`latest-core-version`*`/bin/daemon`, maka daemon sedang berjalan.
   b. Mulai daemon:

```
cd /greengrass/ggc/core/
sudo ./greengrassd start
```

2. Pada halaman konfigurasi grup, pilihDeploy.
3. PadaFungsi Lambdatab, pilihDetektor IPdan pilihlahedit.
4. DariUbah pengaturan detektor IPkotak dialog, pilihSecara otomatis mendeteksi dan mengganti titik akhir broker MQTTdan pilihlahSimpan.

   Hal ini mengaktifkan perangkat untuk secara otomatis memperoleh informasi konektivitas untuk core, seperti alamat IP, DNS, dan nomor port. Deteksi otomatis direkomendasikan, namun AWS IoT Greengrass juga support titik akhir yang ditentukan secara manual. Anda hanya diminta untuk metode penemuan pertama kalinya bahwa grup di-deploy.

   > **Note**
   >
   > Jika diminta, berikan izin untuk membuat Peran layanan Greengrass dan kaitkan dengan Akun AWS Anda pada Wilayah AWS. Peran ini memungkinkan AWS IoT Greengrass untuk mengakses sumber daya Anda di layanan AWS ini.

   Halaman Deployment menampilkan timestamp deployment, ID versi, dan status. Setelah selesai, status yang ditampilkan untuk deployment seharusnyaCompleted (Lengkap).

   Untuk informasi lebih lanjut tentang deployment, lihat Men-deploy AWS IoT Greengrass grup. Untuk bantuan pemecahan masalah, lihat Memecahkan masalah.

Uji contoh inferensi

Sekarang Anda dapat memverifikasi apakah deployment dikonfigurasi dengan benar. Untuk menguji, Anda berlangganan topik `/resnet-50/predictions` dan menerbitkan pesan apa pun ke topik `/resnet-50/test` ini. Ini memicu fungsi Lambda untuk mengambil foto dengan Raspberry Pi Anda dan melakukan inferensi pada gambar yang ditangkap.

> **Note**
>
> Jika menggunakan contoh NVIDIA Jetson, pastikan untuk menggunakan `resnet-18/predictions` dan `resnet-18/test` topik sebagai gantinya.

> **Note**
>
> Jika monitor terpasang ke Raspberry Pi, umpan kamera langsung ditampilkan di jendela pratinjau.

1. PadaAWS IoThalaman beranda konsol, di bawahTEST, chooseKlien uji MQTT.
2. Untuk Langganan, pilih Berlangganan Topik. Gunakan nilai-nilai berikut. Tinggalkan opsi yang tersisa pada defaultnya.
   - Untuk Topik langganan, masukkan **`/resnet-50/predictions`**.
   - Di bawahKonfigurasi tambahan, untukTampilan muatan MQTT, chooseTampilkan muatan sebagai string.
3. Pilih Langganan.
4. PilihMenerbitkan ke topikENTER**`/resnet-50/test`**sebagaiNama topik, dan pilihlahPublikasikan.
5. Jika uji berhasil, pesan diterbitkan menyebabkan kamera Raspberry Pi untuk menangkap citra. Sebuah pesan dari fungsi Lambda muncul di bagian bawah halaman. Pesan ini berisi hasil prediksi citra, menggunakan format: diprediksi nama kelas, probabilitas, dan penggunaan memori puncak.

Mengkonfigurasi Atom Intel

Untuk menjalankan tutorial ini pada perangkat Intel Atom, Anda harus memberikan citra sumber, Mengonfigurasi fungsi Lambda, dan menambahkan sumber daya perangkat lokal lain. Untuk menggunakan GPU sebagai inferensi, pastikan perangkat lunak berikut diinstal pada perangkat Anda:

- OpenCL versi 1.0 atau yang lebih baru
- Python 3.7 dan pip

- NumPy
- OpenCV pada Wheels

1. Unduh citra PNG atau JPG statis untuk fungsi Lambda yang akan digunakan untuk klasifikasi citra. Contoh tersebut bekerja paling baik dengan file citra kecil.

   Simpan file citra Anda di direktori yang memuat file `inference.py` (atau dalam subdirektori dari direktori ini). Hal ini ada dalam paket deployment fungsi Lambda yang Anda unggah di the section called "Buat fungsi inferensi Lambda".

   > **Note**
   >
   > Jika Anda menggunakan AWS DeepLens, Anda dapat menggunakan kamera onboard atau memasang kamera Anda sendiri untuk melakukan inferensi pada citra yang ditangkap, bukan citra statis. Namun, kami sangat merekomendasikan Anda mulai dengan citra statis terlebih dahulu.
   > Jika Anda menggunakan kamera, pastikan bahwa paket `awscam` APT diinstal dan diperbarui. Untuk informasi lebih lanjut, lihat Perbarui Perangkat AWS DeepLens Anda dalam AWS DeepLens Panduan Developer.

2. Edit konfigurasi fungsi Lambda. Ikuti prosedur di the section called "Tambahkan fungsi Lambda ke grup".

   > **Note**
   >
   > Kami merekomendasikan Anda menjalankan fungsi Lambda Anda tanpa kontainerisasi kecuali kasus bisnis Anda memerlukannya. Hal ini membantu mengaktifkan akses ke perangkat GPU dan kamera Anda tanpa mengonfigurasi sumber daya perangkat. Jika Anda menjalankan tanpa kontainerisasi, Anda juga harus memberikan akses root ke Fungsi Lambda AWS IoT Greengrass Anda.

   a. Untuk berjalan tanpa kontainerisasi:

      - UntukPengguna dan grup sistem, choose**`Another user ID/group ID`**. UntukID Pengguna SistemENTER**`0`**. UntukID Grup SistemENTER**`0`**.

Hal ini memungkinkan fungsi Lambda Anda untuk berjalan sebagai root. Untuk informasi lebih lanjut untuk menjalankan sebagai root, lihat the section called “Mengatur identitas akses default untuk fungsi Lambda dalam grup”.

> **Tip**
>
> Anda juga harus memperbarui file `config.json` untuk memberikan akses root ke fungsi Lambda Anda. Untuk prosedur ini, lihat the section called “Menjalankan fungsi Lambda sebagai root”.

- UntukFungsi Lambda kontainerisasi, chooseTanpa kontainer.

  Untuk informasi lebih lanjut tentang berjalan tanpa kontainerisasi, lihat the section called “Pertimbangan ketika memilih fungsi Lambda kontainerisasi”.
- Tingkatkan nilai Timeout untuk 2 menit. Ini memastikan bahwa permintaan batas waktu tidak terlalu pendek. Dibutuhkan beberapa menit setelah pengaturan untuk menjalankan inferensi.
- UntukDipinned, chooseBenar.
- Di bawahParameter tambahan, untukBaca akses ke direktori /sys, chooseDiaktifkan.

b. Untuk menjalankan dalam mode kontainer sebagai gantinya:

> **Note**
>
> Kami tidak merekomendasikan untuk menjalankan dalam mode kontainerisasi kecuali jika kasus bisnis Anda memerlukannya.

- Tingkatkan nilai Batas memori hingga 3000 MB.
- Tingkatkan nilai Timeout untuk 2 menit. Ini memastikan bahwa permintaan batas waktu tidak terlalu pendek. Dibutuhkan beberapa menit setelah pengaturan untuk menjalankan inferensi.
- UntukDipinned, chooseBenar.
- Di bawahParameter tambahan, untukBaca akses ke direktori /sys, chooseDiaktifkan.

3. Tambahkan sumber daya model Neo-dioptimalisasi Anda ke grup. Unggah model sumber daya di direktori `resnet50` dari paket sampel yang Anda unzipped di the section called “Buat fungsi

inferensi Lambda". Direktori ini berisi artefak model precompiled untuk model klasifikasi citra dilatih dengan Resnet-50. Ikuti prosedur di the section called "Tambahkan sumber daya model NEO-dioptimalkan ke grup" dengan pembaruan berikut.

- Kompres file di dalam direktori `resnet50` ke dalam sebuah file bernama `resnet50.zip`.
- Pada halaman Buat sumber daya machine learning ini, untuk Nama sumber daya, masukkan **`resnet50_model`**.
- Unggah `resnet50.zip` file.

4. Jika berjalan dalam mode containerized, tambahkan sumber daya perangkat lokal yang diperlukan untuk memberikan akses ke GPU perangkat Anda.

> **Note**
>
> Jika Anda menjalankan dalam mode nonkontainerisasi, AWS IoT Greengrass dapat mengakses GPU perangkat Anda tanpa mengonfigurasi sumber daya perangkat.

   a. Pada halaman konfigurasi grup, pilihSumber dayaTab.
   b. DiSumber daya lokalbagian, pilihTambahkan sumber daya lokal.
   c. Mendefinisikan sumber daya:
      - Untuk Nama sumber daya, masukkan **`renderD128`**.
      - Untuk Jenis sumber daya, pilih Perangkat.
      - UntukJalur perangkat lokalENTER**`/dev/dri/renderD128`**.
      - UntukPemilik grup sistem dan izin akses file, chooseSecara otomatis menambahkan izin sistem file dari grup sistem yang memiliki sumber daya.

Mengkonfigurasi NVIDIA Jetson TX2

Untuk menjalankan tutorial ini pada NVIDIA Jetson TX2, berikan citra sumber, konfigurasi fungsi Lambda, dan tambahkan lebih banyak sumber daya perangkat lokal.

1. Pastikan perangkat Jetson Anda dikonfigurasi sehingga Anda dapat menginstal AWS IoT Greengrass Perangkat lunak Core dan gunakan GPU untuk inferensi. Untuk informasi lebih lanjut tentang mengonfigurasi Spot Instance Anda, lihat the section called "Mengatur perangkat lain". Untuk menggunakan GPU untuk inferensi pada NVIDIA Jetson TX2, Anda harus menginstal

CUDA 10.0 dan cuDNN 7.0 pada perangkat Anda ketika Anda menggambar papan Anda dengan Jetpack 4.3.

2. Unduh citra PNG atau JPG statis untuk fungsi Lambda yang akan digunakan untuk klasifikasi citra. Contohnya bekerja paling baik dengan file citra kecil.

   Simpan file citra Anda di direktori yang memuat `inference.py` file. Anda juga dapat menyimpannya dalam subdirektori di direktori ini. Direktori ini adalah dalam paket deployment fungsi Lambda yang Anda unggah di the section called “Buat fungsi inferensi Lambda”.

   > **Note**
   >
   > Anda dapat memilih untuk instrumen kamera di papan Jetson untuk menangkap citra sumber. Namun, kami sangat merekomendasikan Anda mulai dengan citra statis terlebih dahulu.

3. Edit konfigurasi fungsi Lambda. Ikuti prosedur di the section called “Tambahkan fungsi Lambda ke grup”.

   > **Note**
   >
   > Kami merekomendasikan Anda menjalankan fungsi Lambda Anda tanpa kontainerisasi kecuali kasus bisnis Anda memerlukannya. Hal ini membantu mengaktifkan akses ke perangkat GPU dan kamera Anda tanpa mengonfigurasi sumber daya perangkat. Jika Anda menjalankan tanpa kontainerisasi, Anda juga harus memberikan akses root ke Fungsi Lambda AWS IoT Greengrass Anda.

   a. Untuk berjalan tanpa kontainerisasi:

      - Untuk Jalankan sebagai, pilih **`Another user ID/group ID`**. Untuk UID, masukkan **`0`**. Untuk GUID, masukkan **`0`**.

        Hal ini memungkinkan fungsi Lambda Anda untuk berjalan sebagai root. Untuk informasi lebih lanjut untuk menjalankan sebagai root, lihat the section called “Mengatur identitas akses default untuk fungsi Lambda dalam grup”.

> **Tip**
>
> Anda juga harus memperbarui file `config.json` untuk memberikan akses root ke fungsi Lambda Anda. Untuk prosedur ini, lihat the section called “Menjalankan fungsi Lambda sebagai root”.

- UntukFungsi Lambda kontainerisasi, chooseTanpa kontainer.

  Untuk informasi lebih lanjut tentang berjalan tanpa kontainerisasi, lihat the section called “Pertimbangan ketika memilih fungsi Lambda kontainerisasi”.
- Tingkatkan nilai Timeout untuk 5 menit. Ini memastikan bahwa permintaan batas waktu tidak terlalu pendek. Dibutuhkan beberapa menit setelah pengaturan untuk menjalankan inferensi.
- UntukDipinned, chooseBenar.
- Di bawahParameter tambahan, untukBaca akses ke direktori /sys, chooseDiaktifkan.

b. Untuk menjalankan dalam mode kontainer sebagai gantinya:

> **Note**
>
> Kami tidak merekomendasikan untuk dijalankan dalam mode kontainer kecuali jika kasus bisnis Anda memerlukannya.

- Tingkatkan nilai batas memori ini. Untuk menggunakan model yang disediakan dalam mode GPU, gunakan setidaknya 2000 MB.
- Tingkatkan nilai Timeout untuk 5 menit. Ini memastikan bahwa permintaan batas waktu tidak terlalu pendek. Dibutuhkan beberapa menit setelah pengaturan untuk menjalankan inferensi.
- UntukDipinned, chooseBenar.
- Di bawahParameter tambahan, untukBaca akses ke direktori /sys, chooseDiaktifkan.

4. Tambahkan sumber daya model Neo-dioptimalisasi Anda ke grup. Unggah model sumber daya di direktori `resnet18` dari paket sampel yang Anda unzipped di the section called “Buat fungsi inferensi Lambda”. Direktori ini berisi artefak model precompiled untuk model klasifikasi citra terlatih dengan Resnet-18. Ikuti prosedur di the section called “Tambahkan sumber daya model NEO-dioptimalkan ke grup” dengan pembaruan berikut.

- Kompres file di dalam direktori `resnet18` ke dalam sebuah file bernama `resnet18.zip`.
- Pada halaman Buat sumber daya machine learning ini, untuk Nama sumber daya, masukkan **`resnet18_model`**.
- Unggah `resnet18.zip` file.

5. Jika berjalan dalam mode containerized, tambahkan sumber daya perangkat lokal yang diperlukan untuk memberikan akses ke GPU perangkat Anda.

> **Note**
>
> Jika Anda berjalan dalam mode nonkontainerisasi, AWS IoT Greengrass dapat mengakses GPU perangkat Anda tanpa mengonfigurasi sumber daya perangkat.

   a. Pada halaman konfigurasi grup, pilihSumber dayaTab.
   b. DiSumber daya lokalbagian, pilihTambahkan sumber daya lokal.
   c. Tentukan setiap sumber daya:
      - Untuk Nama sumber daya dan Jalur perangkat, gunakan nilai-nilai dalam tabel berikut. Buat satu sumber daya perangkat untuk setiap baris dalam tabel.
      - Untuk Jenis sumber daya, pilih Perangkat.
      - UntukPemilik grup sistem dan izin akses file, chooseSecara otomatis menambahkan izin sistem file dari grup sistem yang memiliki sumber daya.

Nama	Jalur perangkat
nvhost-ctrl	/dev/nvhost-ctrl
nvhost-gpu	/dev/nvhost-gpu
nvhost-ctrl-gpu	/dev/dev/nvhost-ctrl-gpu
nvhost-dbg-gpu	/dev/dev/nvhost-dbg-gpu
nvhost-prof-gpu	/dev/dev/nvhost-prof-gpu

Nama	Jalur perangkat
nvmap	/dev/nvmap
nvhost-vic	/dev/nvhost-vic
tegra_dc_ctrl	/dev/tegra_dc_ctrl

6. Jika berjalan dalam mode kontainerisasi, tambahkan sumber daya volume lokal berikut untuk memberikan akses ke kamera perangkat Anda. Ikuti prosedur di the section called “Tambahkan sumber daya model NEO-dioptimalkan ke grup”.

   > **Note**
   >
   > Jika Anda berjalan dalam mode nonkontainerisasi, AWS IoT Greengrass dapat mengakses kamera perangkat Anda tanpa mengonfigurasi sumber daya perangkat.

   - Untuk Jenis sumber daya, pilih Volume.
   - UntukPemilik grup sistem dan izin akses file, chooseSecara otomatis menambahkan izin sistem file dari grup sistem yang memiliki sumber daya.

Nama	Jalur sumber	Jalur tujuan
shm	/dev/shm	/dev/shm
tmp	/tmp	/tmp

7. Memperbarui langganan grup Anda untuk menggunakan direktori yang benar. Ikuti prosedur di the section called “Tambahkan langganan ke grup” dengan pembaruan berikut.

   - Untuk filter topik pertama Anda, masukkan **`/resnet-18/predictions`**.
   - Untuk filter topik kedua Anda, masukkan **`/resnet-18/test`**.

8. Memperbarui langganan pengujian Anda untuk menggunakan direktori yang benar. Ikuti prosedur di the section called “Uji contoh” dengan pembaruan berikut.

- UntukLangganan, chooseBerlangganan topik. Untuk Topik langganan, masukkan **`/resnet-18/predictions`**.
- Pada `/resnet-18/predictions` halaman, tentukan `/resnet-18/test` topik untuk diterbitkan ke.

Pemecahan masalah AWS IoT Greengrass inferensi ML

Jika tes tidak berhasil, Anda dapat mencoba langkah-langkah penyelesaian masalah berikut. Jalankan perintah di terminal Raspberry Pi Anda.

Periksa catatan error

1. Beralih ke pengguna root dan arahkan ke direktori `log` tersebut. Akses ke AWS IoT Greengrass log memerlukan izin root.

   ```
   sudo su
   cd /greengrass/ggc/var/log
   ```

2. Periksa `runtime.log` untuk kesalahan apa pun.

   ```
   cat system/runtime.log | grep 'ERROR'
   ```

   Anda juga dapat melihat di log fungsi Lambda ditetapkan pengguna Anda untuk setiap kesalahan:

   ```
   cat user/your-region/your-account-id/lambda-function-name.log | grep 'ERROR'
   ```

   Untuk informasi selengkapnya, lihat the section called “Pemecahan masalah dengan catatan”.

Verifikasi fungsi Lambda berhasil di-deploy

1. Daftar isi dari Lambda yang di-deploy di `/lambda` direktori. Mengganti nilai placeholder sebelum Anda menjalankan perintah.

```
cd /greengrass/ggc/deployment/lambda/
arn:aws:lambda:region:account:function:function-name:function-version
ls -la
```

2. Verifikasi bahwa direktori berisi konten yang sama sebagai paket deployment `optimizedImageClassification.zip` yang Anda unggah di Langkah 3: Buat fungsi inferensi Lambda.

   Pastikan bahwa `.py` file dan dependensi berada di root direktori.

Verifikasi model inferensi berhasil di-deploy

1. Temukan process identification number (PID) proses waktu aktif Lambda:

   ```
   ps aux | grep lambda-function-name
   ```

   Pada output, PID muncul di kolom kedua baris untuk proses waktu aktif Lambda.

2. Masukkan namespace waktu aktif Lambda. Pastikan untuk mengganti nilai placeholder *pid* sebelum Anda menjalankan perintah.

   > **Note**
   >
   > Direktori ini dan isinya berada di namespace waktu aktif Lambda, sehingga mereka tidak terlihat dalam namespace Linux biasa.

   ```
   sudo nsenter -t pid -m /bin/bash
   ```

3. Daftar isi dari direktori lokal yang Anda tentukan untuk sumber daya ML.

   > **Note**
   >
   > Jika jalur sumber daya ML Anda adalah sesuatu selain `ml_model`, Anda harus mengganti yang di sini.

```
cd /ml_model
ls -ls
```

Anda harus melihat file berikut:

```
    56 -rw-r--r-- 1 ggc_user ggc_group     56703 Oct 29 20:07 model.json
196152 -rw-r--r-- 1 ggc_user ggc_group 200855043 Oct 29 20:08 model.params
   256 -rw-r--r-- 1 ggc_user ggc_group    261848 Oct 29 20:07 model.so
    32 -rw-r--r-- 1 ggc_user ggc_group     30564 Oct 29 20:08 synset.txt
```

Fungsi Lambda **/dev/dri/renderD128**

Hal ini dapat terjadi jika OpenCL tidak dapat terhubung ke perangkat GPU yang dibutuhkannya. Anda harus membuat sumber daya perangkat untuk perangkat yang diperlukan untuk fungsi Lambda Anda.

Langkah selanjutnya

Selanjutnya, jelajahi model lain yang dioptimalkan. Untuk informasi, lihatSageMaker Dokumentasi Neo.

Mengelola aliran data di AWS IoT Greengrass core

AWS IoT Greengrass pengelola pengaliran membuatnya lebih mudah dan lebih dapat diandalkan untuk mentransfer data IoT volume tinggi ke AWS Cloud. Pengelola pengaliran memproses pengaliran data secara lokal dan mengekspornya ke AWS Cloud secara otomatis. Fitur ini terintegrasi dengan skenario edge umum, seperti inferensi machine learning (ML), di mana data diproses dan dianalisis secara lokal sebelum diekspor ke AWS Cloud atau tujuan penyimpanan lokal.

Pengelola pengaliran memudahkan pengembangan aplikasi. Aplikasi IoT Anda dapat menggunakan mekanisme standar untuk memproses volume tinggi pengaliran dan mengelola kebijakan penyimpanan data lokal bukannya membangun fungsionalitas manajemen pengaliran kustom. Aplikasi IoT dapat membaca dan menulis ke aliran. Mereka dapat menentukan kebijakan untuk jenis penyimpanan, ukuran, dan retensi data per pengaliran untuk mengontrol cara pengelola pengaliran memproses dan mengekspor pengaliran.

Pengelola pengaliran dirancang untuk bekerja di lingkungan dengan konektivitas intermiten atau terbatas. Anda dapat menentukan penggunaan bandwidth, perilaku timeout, dan cara aliran data ditangani ketika core terhubung atau terputus. Untuk data penting, Anda dapat menetapkan prioritas untuk mengontrol urutan pengaliran yang diekspor ke AWS Cloud.

Anda dapat mengonfigurasi ekspor otomatis ke AWS Cloud untuk penyimpanan atau pemrosesan dan analisis lebih lanjut. Pengelola pengaliran mendukung mengekspor ke AWS Cloud tujuan.

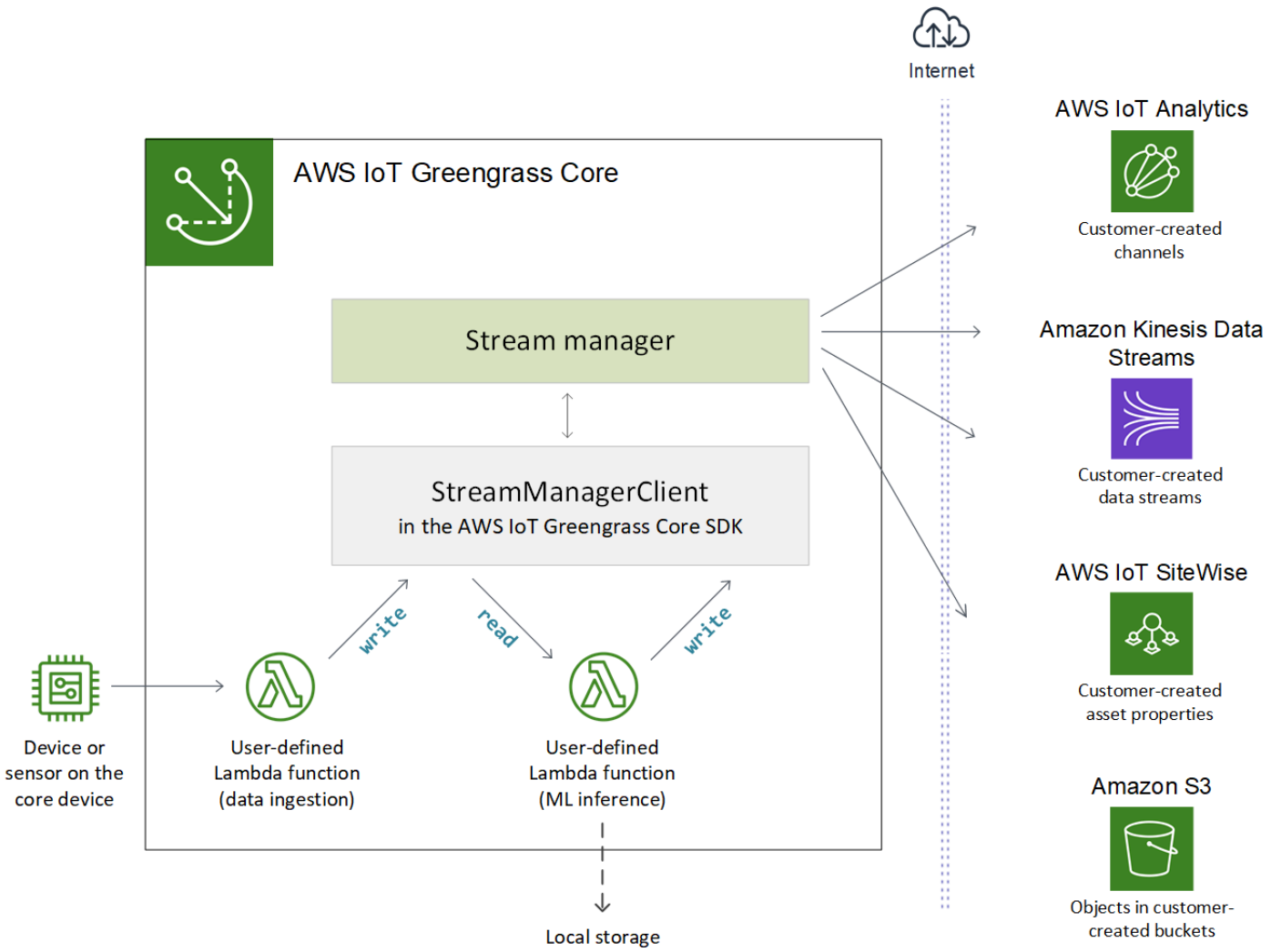
- Saluran di AWS IoT Analytics. AWS IoT Analytics mengizinkan Anda melakukan analisis lanjutan pada data Anda untuk membantu membuat keputusan bisnis dan meningkatkan model machine learning. Untuk informasi lebih lanjut, lihat Apa AWS IoT Analytics? dalam AWS IoT Analytics Panduan Pengguna.
- Aliran di Kinesis Data Streams. Kinesis Data Streams umumnya digunakan untuk mengumpulkan data volume tinggi dan memuatnya ke dalam gudang data atau klaster map-reduce. Untuk informasi lebih lanjut, lihat Apa itu Amazon Kinesis Data Streams? dalam Panduan Developer Amazon Kinesis.
- Properti aset di AWS IoT SiteWise. AWS IoT SiteWise mengizinkan Anda mengumpulkan, mengatur, dan menganalisis data dari peralatan industri dalam menskalakan. Untuk informasi lebih lanjut, lihat Apa AWS IoT SiteWise? dalam AWS IoT SiteWise Panduan Pengguna.
- Objek dalam Amazon S3. Anda dapat menggunakan Amazon S3 untuk menyimpan dan mengambil data dalam jumlah besar. Untuk informasi lebih lanjut, lihat Apa Amazon S3? dalam Panduan Developer Amazon Simple Storage Service.

alur kerja manajemen pengaliran

Aplikasi IoT Anda berinteraksi dengan pengelola pengaliran melalui AWS IoT Greengrass Core SDK. Dalam alur kerja yang sederhana, fungsi Lambda yang ditetapkan pengguna berjalan pada core Greengrass mengonsumsi data IoT, seperti suhu time-series dan tekanan metrik. Fungsi Lambda mungkin memfilter atau mengompres data lalu memanggil AWS IoT Greengrass Core SDK untuk menulis data ke pengaliran di pengelola pengaliran. Pengelola pengaliran dapat mengekspor pengaliran ke AWS Cloud secara otomatis, berdasarkan kebijakan yang ditetapkan untuk pengaliran. Fungsi Lambda yang ditetapkan pengguna juga dapat mengirim data langsung ke basis data lokal atau repositori penyimpanan.

Aplikasi IoT Anda dapat mencakup beberapa fungsi Lambda yang ditetapkan pengguna yang membaca atau menulis ke pengaliran. Ini fungsi Lambda lokal dapat membaca dan menulis ke pengaliran untuk memfilter, agregat, dan menganalisis data lokal. Hal ini mengizinkan untuk merespons dengan cepat peristiwa lokal dan mengekstraksi informasi berharga sebelum data ditransfer dari core ke cloud atau tujuan lokal.

Contoh alur kerja seperti yang ditunjukkan dalam diagram berikut.



Untuk menggunakan pengelola pengaliran, mulai dengan mengonfigurasi parameter pengelola pengaliran untuk menentukan pengaturan waktu aktif tingkat grup yang berlaku untuk semua pengaliran pada core Greengrass. Pengaturan yang dapat disesuaikan ini mengizinkan Anda mengontrol cara pengelola pengaliran menyimpan, memproses, dan mengekspor pengaliran berdasarkan kebutuhan bisnis dan kendala lingkungan Anda. Untuk informasi lebih lanjut, lihat the section called “Konfigurasi pengelola pengaliran”.

Setelah Anda mengonfigurasi pengelola pengaliran, Anda dapat membuat dan men-deploy aplikasi IoT Anda. Ini biasanya ditetapkan pengguna fungsi Lambda yang menggunakan `StreamManagerClient` di AWS IoT Greengrass Core SDK untuk membuat dan berinteraksi dengan pengaliran. Selama pembuatan pengaliran, fungsi Lambda mendefinisikan kebijakan per-pengaliran, seperti tujuan ekspor, prioritas, dan ketekunan. Untuk informasi lebih lanjut, termasuk cuplikan kode untuk `StreamManagerClient` operasi, lihat the section called “Gunakan StreamManagerClient untuk bekerja dengan aliran”.

Untuk tutorial yang mengonfigurasi alur kerja sederhana, lihat the section called “Ekspor aliran data (konsol)” atau the section called “Ekspor aliran data (CLI)”.

Persyaratan

Persyaratan berikut berlaku untuk menggunakan pengelola pengaliran:

- Anda harus menggunakan AWS IoT Greengrass perangkat lunak Core v1.10 atau yang lebih baru, dengan pengelola pengaliran diaktifkan. Untuk informasi selengkapnya, lihat the section called “Konfigurasi pengelola pengaliran”.

  Pengelola pengaliran tidak didukung pada OpenWrt distribusi.
- Waktu aktif Java 8 (JDK 8) harus diinstal pada core.
  - Untuk distribusi berbasis Debian (termasuk Raspbian) atau distribusi berbasis Ubuntu, jalankan perintah berikut:

    ```
    sudo apt install openjdk-8-jdk
    ```
  - Untuk distribusi berbasis Red Hat (termasuk Amazon Linux), jalankan perintah berikut:

    ```
    sudo yum install java-1.8.0-openjdk
    ```

    Untuk informasi lebih lanjut, lihat Cara mengunduh dan menginstal paket OpenJDK prebuilt dalam dokumentasi OpenJDK.

- Pengelola pengaliran membutuhkan RAM minimal 70 MB selain basis Anda AWS IoT Greengrass perangkat Core. Kebutuhan memori total Anda tergantung pada beban kerja Anda.

- Fungsi Lambda yang ditetapkan pengguna harus menggunakan AWS IoT Greengrass Core SDK untuk berinteraksi dengan pengelola pengaliran. Core SDK AWS IoT Greengrass tersedia dalam beberapa bahasa, tetapi hanya versi berikut yang mendukung operasi pengelola pengaliran:
  - Java SDK (v1.4.0 atau yang lebih baru)
  - Python SDK (v1.5.0 atau yang lebih baru)
  - Node.js SDK (v1.6.0 atau yang lebih baru)

Unduh versi SDK yang sesuai dengan fungsi waktu aktif Lambda Anda dan memasukkannya ke dalam paket deployment fungsi Lambda Anda.

> **Note**
>
> Core SDK for Python AWS IoT Greengrass membutuhkan Python 3.7 atau yang lebih baru dan memiliki dependensi paket lainnya. Untuk informasi lebih lanjut, lihat Buat paket deployment fungsi Lambda (konsol) atau Buat paket deployment fungsi Lambda (CLI).

- Jika Anda menentukan AWS Cloud ekspor tujuan untuk pengaliran, Anda harus membuat target ekspor Anda dan memberikan izin mengakses dalam peran grup Greengrass. Tergantung pada tujuan, persyaratan lain mungkin juga berlaku. Untuk informasi selengkapnya, lihat:
  - the section called “AWS IoT Analytics saluran”
  - the section called “Amazon Kinesis data streams”
  - the section called “Properti aset AWS IoT SiteWise”
  - the section called “Objek Amazon S3”

  Anda bertanggung jawab untuk menjaga sumber daya AWS Cloud ini.

Keamanan data

Bila Anda menggunakan stream manager, perhatikan pertimbangan keamanan berikut ini.

Keamanan data lokal

AWS IoT Greengrass tidak mengenkripsi data at rest pengaliran atau transit secara lokal antara komponen pada perangkat core.

- Data at rest. Data pengaliran disimpan secara lokal dalam direktori penyimpanan pada core Greengrass. Untuk keamanan data, AWS IoT Greengrass bergantung pada izin file Unix dan enkripsi disk penuh, jika diaktifkan. Anda dapat menggunakan parameter opsional STREAM_MANAGER_STORE_ROOT_DIR untuk menentukan direktori penyimpanan. Jika Anda mengubah parameter ini nanti untuk menggunakan direktori penyimpanan yang berbeda, AWS IoT Greengrass tidak menghapus direktori penyimpanan sebelumnya atau isinya.

- Data dalam transit lokal. AWS IoT Greengrass tidak mengenkripsi data pengaliran dalam transit lokal pada core antara sumber data, fungsi Lambda, AWS IoT Greengrass Core SDK, dan pengelola pengaliran.

- Data dalam transit ke AWS Cloud. Aliran data yang diekspor oleh pengelola pengaliran ke AWS Cloud gunakan standar AWS layanan enkripsi klien dengan Transport Layer Security (TLS).

Untuk informasi selengkapnya, lihat the section called “Enkripsi data”.

Autentikasi klien

Klien pengelola pengaliran menggunakan AWS IoT Greengrass core SDK untuk berkomunikasi dengan pengelola pengaliran. Ketika autentikasi klien diaktifkan, hanya fungsi Lambda dalam grup Greengrass dapat berinteraksi dengan pengaliran di pengelola pengaliran. Ketika autentikasi klien dinonaktifkan, setiap proses yang berjalan pada core Greengrass (seperti kontainer Docker) dapat berinteraksi dengan pengaliran di pengelola pengaliran. Anda harus menonaktifkan autentikasi hanya jika kasus bisnis Anda memerlukannya.

Anda menggunakan STREAM_MANAGER_AUTHENTICATE_CLIENT parameter untuk mengatur mode autentikasi klien. Anda dapat mengonfigurasi parameter ini dari konsol atau AWS IoT Greengrass API. Perubahan berlaku setelah grup di-deploy.

	Diaktifkan	Dinonaktifkan
Nilai parameter	`true` (default dan disarankan)	`false`
Klien yang diizinkan	Fungsi Lambda yang ditetapkan pengguna dalam grup Greengrass	Fungsi Lambda yang ditetapkan pengguna dalam grup Greengrass Proses lain yang berjalan di perangkat core Greengrass

Lihat juga

- the section called “Konfigurasi pengelola pengaliran”

- the section called “Gunakan StreamManagerClient untuk bekerja dengan aliran”
- the section called “Konfigurasi ekspor untuk tujuan AWS Cloud yang didukung”
- the section called “Ekspor aliran data (konsol)”
- the section called “Ekspor aliran data (CLI)”

Konfigurasikan manajer pengaliran AWS IoT Greengrass

Pada AWS IoT Greengrass core, pengelola pengaliran dapat menyimpan, memproses, dan mengekspor data perangkat IoT. Pengelola pengaliran menyediakan parameter yang Anda gunakan untuk mengonfigurasi waktu aktif pengaturan tingkat grup. Pengaturan ini berlaku untuk semua pengaliran pada core Greengrass. Anda dapat menggunakan konsol AWS IoT tersebut atau AWS IoT Greengrass API untuk mengonfigurasi pengaturan pengelola pengaliran. Perubahan berlaku setelah grup di-deploy.

> **Note**
>
> Setelah Anda mengonfigurasi pengelola pengaliran, Anda dapat membuat dan men-deploy aplikasi IoT yang berjalan pada core Greengrass dan berinteraksi dengan pengelola pengaliran. Aplikasi IoT ini biasanya ditetapkan pengguna fungsi Lambda. Untuk informasi selengkapnya, lihat the section called “Gunakan StreamManagerClient untuk bekerja dengan aliran”.

Parameter pengelola pengaliran

Pengelola pengaliran menyediakan parameter berikut yang mengizinkan Anda untuk menentukan pengaturan tingkat grup. Semua parameter bersifat opsional.

Penyimpanan direktori

Nama parameter: `STREAM_MANAGER_STORE_ROOT_DIR`

Path absolut dari direktori lokal yang digunakan untuk menyimpan pengaliran. Nilai ini harus dimulai dengan garis miring ke depan (misalnya, `/data`).

Untuk informasi tentang cara mengamankan data aliran, lihat the section called “Keamanan data lokal”.

MinimumAWS IoT GreengrassVersi inti: 1.10.0 0

Port server

Nama parameter: `STREAM_MANAGER_SERVER_PORT`

Nomor port lokal yang digunakan untuk berkomunikasi dengan stream manager. Default-nya adalah `8088`.

MinimumAWS IoT GreengrassVersi inti: 1.10.0 0

Autentikasi klien

Nama parameter: `STREAM_MANAGER_AUTHENTICATE_CLIENT`

Mengindikasikan apakah klien harus diautentikasi untuk berinteraksi dengan pengelola pengaliran. Semua interaksi antara klien dan pengelola pengaliran dikendalikan oleh AWS IoT Greengrass Core SDK. Parameter ini menentukan klien yang dapat memanggil AWS IoT Greengrass Core SDK untuk bekerja dengan pengaliran. Untuk informasi selengkapnya, lihat the section called “Autentikasi klien”.

Nilai yang valid adalah `true` atau `false`. Default-nya adalah `true` (direkomendasikan).

- `true`. Mengizinkan hanya fungsi Greengrass Lambda sebagai klien. Klien fungsi Lambda menggunakan internal AWS IoT Greengrass protokol core untuk mengautentikasi dengan AWS IoT Greengrass Core SDK.
- `false`. Mengizinkan setiap proses yang berjalan pada AWS IoT Greengrass core untuk menjadi klien. Jangan diatur ke `false` kecuali kasus bisnis Anda memerlukannya. Sebagai contoh, atur nilai ini ke `false` hanya jika proses non-Lambda pada perangkat core harus berkomunikasi langsung dengan pengelola pengaliran, seperti kontainer Docker berjalan pada core.

MinimumAWS IoT GreengrassVersi inti: 1.10.0 0

Bandwidth maksimum

Nama parameter: `STREAM_MANAGER_EXPORTER_MAX_BANDWIDTH`

Bandwidth maksimum rata-rata (dalam kilobit per detik) yang dapat digunakan untuk mengekspor data. Default ini memungkinkan penggunaan bandwidth yang tersedia tanpa batas.

MinimumAWS IoT GreengrassVersi inti: 1.10.0 0

Ukuran kolam utas

Nama parameter: `STREAM_MANAGER_EXPORTER_THREAD_POOL_SIZE`

Jumlah maksimum utas aktif yang dapat digunakan untuk mengekspor data. Default-nya adalah 5.

Ukuran optimal tergantung pada perangkat keras Anda, volume aliran, dan jumlah yang direncanakan dari aliran ekspor. Jika kecepatan ekspor lambat, Anda dapat menyesuaikan pengaturan ini untuk menemukan ukuran optimal untuk perangkat keras dan kasus bisnis Anda. CPU dan memori perangkat keras inti Anda merupakan faktor pembatas. Untuk memulai, Anda dapat mencoba menetapkan nilai ini sama dengan jumlah inti prosesor pada perangkat.

Hati-hati untuk tidak menetapkan ukuran yang lebih tinggi dari yang dapat didukung perangkat keras Anda. Setiap pengaliran mengonsumsi sumber daya perangkat keras, sehingga Anda harus mencoba untuk membatasi jumlah pengaliran ekspor pada perangkat dibatasi.

MinimumAWS IoT GreengrassVersi inti: 1.10.0 0

Argumen JVM

Nama parameter: `JVM_ARGS`

Argumen Mesin Virtual Java kustom yang akan disampaikan ke stream manager saat startup. Beberapa argumen harus dipisahkan oleh spasi.

Gunakan parameter ini hanya ketika Anda harus menimpa pengaturan default yang digunakan oleh JVM. Misalnya, Anda mungkin perlu meningkatkan ukuran timbunan default jika berencana mengekspor sejumlah besar pengaliran.

MinimumAWS IoT GreengrassVersi inti: 1.10.0 0

Direktori file input baca-saja

Nama parameter: `STREAM_MANAGER_READ_ONLY_DIRS`

Daftar dipisahkan koma jalur absolut ke direktori di luar sistem file root yang menyimpan file input. Pengelola pengaliran membaca dan mengunggah file ke Amazon S3 dan mount direktori sebagai baca-saja. Untuk informasi lebih lanjut tentang mengekspor ke Amazon S3, lihat the section called “Objek Amazon S3”.

Gunakan parameter ini hanya jika kondisi berikut benar:

- Direktori file input untuk pengaliran yang ekspor ke Amazon S3 adalah di salah satu lokasi berikut:
  - Partisi selain sistem file root.
  - Di bawah `/tmp` pada sistem file root.
- Sebuah Kontainerisasi default dari grup Greengrass adalah kontainer Greengrass.

Nilai contoh: `/mnt/directory-1,/mnt/directory-2,/tmp`

MinimumAWS IoT GreengrassVersi inti: 1.11.0 0

Ukuran minimum untuk unggahan multipart

Nama parameter: `STREAM_MANAGER_EXPORTER_S3_DESTINATION_MULTIPART_UPLOAD_MIN_PART_SIZE_BYTES`

Ukuran minimum (dalam byte) dari bagian dalam unggahan multipart ke Amazon S3. Stream manager menggunakan pengaturan ini dan ukuran file inputnya untuk menentukan bagaimana melakukan batch data dalam permintaan PUT multipart. Nilai minimum dan default adalah `5242880` byte (5 MB).

> **Note**
>
> Pengelola pengaliran menggunakan `sizeThresholdForMultipartUploadBytes` properti untuk menentukan apakah akan mengekspor ke Amazon S3 sebagai unggahan multipart atau tunggal. Fungsi Lambda yang ditentukan pengguna menetapkan ambang batas ini ketika mereka membuat pengaliran yang mengekspor ke Amazon S3. Ambang batas default adalah 5 MB.

MinimumAWS IoT GreengrassVersi inti: 1.11.0 0

Konfigurasikan pengaturan pengelola pengaliran (konsol)

Anda dapat menggunakan AWS IoT konsol untuk tugas-tugas manajemen berikut:

- Periksa apakah pengelola pengaliran diaktifkan
- Mengaktifkan atau menonaktifkan pengelola pengaliran selama pembuatan grup
- Aktifkan atau Nonaktifkan pengelola pengaliran untuk grup yang ada
- Ubah pengaturan pengelola pengaturan

Perubahan berlaku setelah grup Greengrass di-deploy. Untuk tutorial yang menunjukkan cara untuk men-deploy grup Greengrass yang berisi fungsi Lambda yang berinteraksi dengan pengelola pengaliran, lihat the section called “Ekspor aliran data (konsol)”.

> **Note**
>
> Ketika Anda menggunakan konsol untuk mengaktifkan pengelola pengaliran dan men-deploy grup, ukuran memori untuk pengelola pengaliran diatur ke 4194304 KB (4 GB) secara default. Kami merekomendasikan Anda mengatur ukuran memori ke setidaknya 128000 KB.

Untuk memeriksa apakah pengelola pengaliran diaktifkan (konsol)

1. DiAWS IoTpanel navigasi konsol, di bawahKelola, PerluasPerangkat Greengrass, dan kemudian pilihGrup (V1).
2. Pilih grup target.
3. PilihTab fungsi Lambda.
4. Di bawahFungsi Lambda sistem, pilihManajer pengaliran, lalu pilihedit.
5. Periksa status diaktifkan atau dinonaktifkan. Pengaturan pengelola pengaliran kustom yang dikonfigurasi juga ditampilkan.

Untuk mengaktifkan atau menonaktifkan pengelola pengaliran selama pembuatan grup (konsol)

1. DiAWS IoTpanel navigasi konsol, di bawahKelola, PerluasPerangkat Greengrass, dan kemudian pilihGrup (V1).
2. Pilih Buat Grup. Pilihan Anda pada halaman selanjutnya menentukan cara mengonfigurasi pengelola pengaliran untuk grup.
3. Lanjutkan melaluiBeri Nama Grup Andalalu pilihCore Greengrasshalaman.
4. Pilih Create group (Buat grup).
5. Pada halaman konfigurasi grup, pilih tombolFungsi Lambdatab, pilihManajer pengaliran, lalu pilihedit.

- Untuk mengaktifkan pengelola pengaliran dengan pengaturan default, pilihAktifkan dengan pengaturan default.

- Untuk mengaktifkan pengelola pengaliran dengan pengaturan kustom, pilih Kustomkan pengaturan.
  1. PadaKonfigurasikan manajer pengaliranhalaman, pilihAktifkan dengan pengaturan kustom.
  2. Di bawah Pengaturan kustom, masukkan nilai untuk parameter pengelola pengaliran. Untuk informasi selengkapnya, lihat the section called “Parameter pengelola pengaliran”. Biarkan bidang kosong untuk mengizinkan AWS IoT Greengrass untuk menggunakan nilai default mereka.

- Untuk menonaktifkan pengelola pengaliran, pilihNonaktifkan.
  1. Pada halaman Konfigurasi pengelola pengaliran ini, pilih Nonaktifkan.

6. Pilih Save (Simpan).
7. Lanjutkan melalui halaman yang tersisa untuk membuat grup Anda.
8. PadaPerangkat klienhalaman, unduh sumber daya keamanan Anda, tinjau informasi, lalu pilihSelesai.

> **Note**
>
> Ketika pengelola pengaliran diaktifkan, Anda harus menginstal waktu aktif Java 8 pada perangkat core sebelum Anda men-deploy grup.

Untuk mengaktifkan atau menonaktifkan pengelola pengaliran untuk grup yang sudah ada (konsol)

1. DiAWS IoTpanel navigasi konsol, di bawahKelola, PerluasPerangkat Greengrass, dan kemudian pilihGrup (V1).
2. Pilih grup target.

3. PilihTab fungsi Lambda.
4. Di bawahFungsi Lambda sistem, pilihManajer pengaliran, lalu pilihedit.
5. Periksa status diaktifkan atau dinonaktifkan. Pengaturan pengelola pengaliran kustom yang dikonfigurasi juga ditampilkan.

Untuk mengubah pengaturan pengelola pengaliran (konsol)

1. DiAWS IoTpanel navigasi konsol, di bawahKelola, PerluasPerangkat Greengrass, dan kemudian pilihGrup (V1).
2. Pilih grup target.
3. PilihTab fungsi Lambda.
4. Di bawahFungsi Lambda sistem, pilihManajer pengaliran, lalu pilihedit.
5. Periksa status diaktifkan atau dinonaktifkan. Pengaturan pengelola pengaliran kustom yang dikonfigurasi juga ditampilkan.
6. Pilih Save (Simpan).

Konfigurasi pengaturan pengelola pengaliran (CLI)

Dalam AWS CLI, gunakan sistem `GGStreamManager` fungsi Lambda untuk mengonfigurasi pengelola pengaliran. Fungsi sistem Lambda adalah komponen dari AWS IoT Greengrass perangkat Core. Untuk pengelola pengaliran dan beberapa fungsi sistem Lambda lainnya, Anda dapat mengonfigurasi fungsi Greengrass dengan mengelola yang sesuai `Function` dan `FunctionDefinitionVersion` objek dalam grup Greengrass. Untuk informasi selengkapnya, lihat the section called “Ringkasan tentang model objek grup”.

Anda dapat menggunakan API untuk tugas-tugas manajemen berikut. Contoh dalam bagian ini menunjukkan cara menggunakan AWS CLI, tetapi Anda juga dapat memanggil AWS IoT Greengrass API secara langsung atau menggunakan AWS SDK.

- Periksa apakah pengelola pengaliran diaktifkan
- Mengaktifkan, menonaktifkan, atau mengonfigurasi pengelola pengaliran

Perubahan berlaku setelah grup di-deploy. Untuk tutorial yang menunjukkan cara untuk men-deploy grup Greengrass dengan fungsi Lambda yang berinteraksi dengan pengelola pengaliran, lihat the section called “Ekspor aliran data (CLI)”.

> **Tip**
>
> Untuk melihat apakah pengelola pengaliran diaktifkan dan berjalan dari perangkat core Anda, Anda dapat menjalankan perintah berikut di dalam terminal pada perangkat.
>
> ```
> ps aux | grep -i 'streammanager'
> ```

Untuk memeriksa apakah pengelola pengaliran diaktifkan (CLI)

Pengelola pengaliran diaktifkan jika versi definisi fungsi yang di-deploy Anda mencakup `GGStreamManager` fungsi Lambda sistem. Untuk memeriksa, lakukan hal berikut;

1. Dapatkan ID dari grup Greengrass target dan versi grup. Prosedur ini mengasumsikan bahwa ini adalah versi grup dan grup terbaru. Query berikut mengembalikan grup yang paling baru dibuat.

   ```
   aws greengrass list-groups --query "reverse(sort_by(Groups, &CreationTimestamp))
   [0]"
   ```

   Atau, Anda dapat melakukan query berdasarkan nama. Nama grup tidak perlu unik, sehingga beberapa grup mungkin ditampilkan.

   ```
   aws greengrass list-groups --query "Groups[?Name=='MyGroup']"
   ```

   > **Note**
   >
   > Anda juga dapat menemukan nilai-nilai ini di konsol AWS IoT tersebut. ID grup ditampilkan pada halaman Pengaturan grup. ID versi grup ditampilkan pada grupDeploymenttab.

2. Salin nilai `Id` dan `LatestVersion` dari grup target di dalam output.
3. Dapatkan versi grup terbaru.

- Ganti *id-grup* dengan `Id` yang Anda salin.
- Ganti*latest-group-version-id*dengan`LatestVersion`yang Anda salin.

```
aws greengrass get-group-version \
--group-id group-id \
--group-version-id latest-group-version-id
```

4. Dari `FunctionDefinitionVersionArn` di dalam output, dapatkan ID dari definisi fungsi dan versi definisi fungsi.

   - ID definisi fungsi adalah GUID yang mengikuti `functions` segmen di dalam Amazon Resource Name (ARN).
   - ID versi definisi fungsi adalah GUID yang mengikuti `versions` segmen di dalam ARN.

   ```
   arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/
   functions/function-definition-id/versions/function-definition-version-id
   ```

5. Dapatkan versi definisi fungsi.

   - Ganti*function-definition-id*dengan ID definisi fungsi.
   - Ganti*function-definition-version-id*dengan ID versi definisi fungsi.

   ```
   aws greengrass get-function-definition-version \
   --function-definition-id function-definition-id \
   --function-definition-version-id function-definition-version-id
   ```

Jika `functions` array di dalam output termasuk `GGStreamManager` fungsi, maka pengelola pengaliran diaktifkan. Setiap variabel lingkungan yang didefinisikan untuk fungsi mewakili pengaturan kustom untuk pengelola pengaliran.

Untuk mengaktifkan, menonaktifkan, atau mengonfigurasi pengelola pengaliran (CLI)

Dalam AWS CLI, gunakan sistem `GGStreamManager` fungsi Lambda untuk mengonfigurasi pengelola pengaliran. Perubahan berlaku setelah Anda men-deploy grup.

- Untuk mengaktifkan pengelola pengaliran, sertakan `GGStreamManager` di dalam `functions` array versi definisi fungsi Anda. Untuk mengonfigurasi pengaturan kustom, definisikan variabel lingkungan yang sesuai untuk parameter pengelola pengaliran.
- Untuk menonaktifkan pengelola pengaliran, hapus `GGStreamManager` dari `functions` array versi definisi fungsi Anda.

Pengelola pengaliran dengan pengaturan default

Contoh konfigurasi berikut mengaktifkan pengelola pengaliran dengan pengaturan default. Ini menetapkan ID fungsi secara paksa ke `streamManager`.

```
{
    "FunctionArn": "arn:aws:lambda:::function:GGStreamManager:1",
    "FunctionConfiguration": {
        "MemorySize": 4194304,
        "Pinned": true,
        "Timeout": 3
    },
    "Id": "streamManager"
}
```

> **Note**
>
> Untuk properti `FunctionConfiguration` ini, Anda mungkin mengetahui hal berikut:
>
> - `MemorySize` diatur ke 4194304 KB (4 GB) dengan pengaturan default. Anda selalu dapat mengubah nilai ini. Kami menganjurkan Anda untuk mengatur `MemorySize` ke setidaknya 128000 KB.
> - `Pinned` harus diatur ke `true`.
> - `Timeout` diperlukan oleh versi definisi fungsi, tetapi `GGStreamManager` tidak menggunakannya.

Pengelola pengaliran dengan pengaturan kustom

Contoh konfigurasi berikut mengaktifkan manajer pengaliran dengan nilai-nilai kustom untuk direktori penyimpanan, server port, dan parameter ukuran kolam thread.

```
{
```

```
    "FunctionArn": "arn:aws:lambda:::function:GGStreamManager:1",
    "FunctionConfiguration": {
        "Environment": {
            "Variables": {
                "STREAM_MANAGER_STORE_ROOT_DIR": "/data",
                "STREAM_MANAGER_SERVER_PORT": "1234",
                "STREAM_MANAGER_EXPORTER_THREAD_POOL_SIZE": "4"
            }
        },
        "MemorySize": 4194304,
        "Pinned": true,
        "Timeout": 3
    },
    "Id": "streamManager"
}
```

AWS IoT Greengrass menggunakan nilai default untuk parameter pengelola pengaliran yang tidak ditentukan sebagai variabel lingkungan.

Pengelola pengaliran dengan pengaturan kustom untuk ekspor Amazon S3

Contoh konfigurasi berikut mengaktifkan pengelola pengaliran dengan nilai-nilai kustom untuk direktori unggah dan parameter ukuran unggah multi-bagian minimum.

```
{
    "FunctionArn": "arn:aws:lambda:::function:GGStreamManager:1",
    "FunctionConfiguration": {
        "Environment": {
            "Variables": {
                "STREAM_MANAGER_READ_ONLY_DIRS": "/mnt/directory-1,/mnt/
directory-2,/tmp",

 "STREAM_MANAGER_EXPORTER_S3_DESTINATION_MULTIPART_UPLOAD_MIN_PART_SIZE_BYTES":
 "10485760"
            }
        },
        "MemorySize": 4194304,
        "Pinned": true,
        "Timeout": 3
    },
    "Id": "streamManager"
}
```

Untuk mengaktifkan, menonaktifkan, atau mengonfigurasi pengelola pengaliran (CLI)

1. Dapatkan ID dari grup Greengrass target dan versi grup. Prosedur ini mengasumsikan bahwa ini adalah versi grup dan grup terbaru. Query berikut mengembalikan grup yang paling baru dibuat.

   ```
   aws greengrass list-groups --query "reverse(sort_by(Groups, &CreationTimestamp))
   [0]"
   ```

   Atau, Anda dapat melakukan query berdasarkan nama. Nama grup tidak perlu unik, sehingga beberapa grup mungkin ditampilkan.

   ```
   aws greengrass list-groups --query "Groups[?Name=='MyGroup']"
   ```

   > **Note**
   >
   > Anda juga dapat menemukan nilai-nilai ini di konsol AWS IoT tersebut. ID grup ditampilkan pada halaman Pengaturan grup. ID versi grup ditampilkan pada grupDeploymenttab.

2. Salin nilai `Id` dan `LatestVersion` dari grup target di dalam output.
3. Dapatkan versi grup terbaru.

   - Ganti *id-grup* dengan `Id` yang Anda salin.
   - Ganti*latest-group-version-id*dengan`LatestVersion`yang Anda salin.

   ```
   aws greengrass get-group-version \
   --group-id group-id \
   --group-version-id latest-group-version-id
   ```

4. Salin `CoreDefinitionVersionArn` dan semua ARN versi lain dari output, kecuali `FunctionDefinitionVersionArn`. Anda menggunakan nilai-nilai ini nanti ketika membuat versi grup.
5. Dari `FunctionDefinitionVersionArn` di dalam output, salin ID dari definisi fungsi. ID adalah GUID yang mengikuti `functions` segmen di dalam ARN, seperti yang ditunjukkan dalam contoh berikut.

```
arn:aws:greengrass:us-west-2:123456789012:/greengrass/
definition/functions/bcfc6b49-beb0-4396-b703-6dEXAMPLEcu5/
versions/0f7337b4-922b-45c5-856f-1aEXAMPLEsf6
```

> **Note**
>
> Atau, Anda dapat membuat definisi fungsi dengan menjalankan `create-function-definition` perintah, dan kemudian menyalin ID dari output.

6. Menambahkan versi definisi fungsi untuk definisi fungsi.

   - Ganti*`function-definition-id`*dengan`Id`yang Anda salin untuk definisi fungsi.
   - Di dalam `functions` array, masukkan semua fungsi lain yang Anda inginkan tersedia pada core Greengrass. Anda dapat menggunakan perintah `get-function-definition-version` untuk mendapatkan daftar fungsi yang ada.

   Aktifkan pengelola pengaliran dengan pengaturan default

   Contoh berikut mengaktifkan pengelola pengaliran, dengan memasukkan `GGStreamManager` di `functions` array. Contoh ini menggunakan nilai default untuk parameter pengelola pengaliran.

```
aws greengrass create-function-definition-version \
--function-definition-id function-definition-id \
--functions '[
        {
            "FunctionArn": "arn:aws:lambda:::function:GGStreamManager:1",
            "FunctionConfiguration": {
                "MemorySize":  4194304,
                "Pinned": true,
                "Timeout": 3
            },
            "Id": "streamManager"
        },
        {
            "FunctionArn": "arn:aws:lambda:us-
west-2:123456789012:function:MyLambdaFunction:MyAlias",
```

```
            "FunctionConfiguration": {
                "Executable": "myLambdaFunction.function_handler",
                "MemorySize": 16000,
                "Pinned": true,
                "Timeout": 5
            },
            "Id": "myLambdaFunction"
        },
        ... more user-defined functions
    ]
}'
```

> **Note**
>
> fungsi `myLambdaFunction` di dalam contoh mewakili salah satu fungsi Lambda yang ditetapkan pengguna Anda.

Aktifkan pengelola pengaliran dengan pengaturan kustom

Contoh berikut mengaktifkan pengelola pengaliran dengan memasukkan fungsi `GGStreamManager` di dalam `functions` array. Semua pengaturan pengelola pengaliran bersifat opsional, kecuali jika Anda ingin mengubah nilai default. Contoh ini menunjukkan cara menggunakan variabel lingkungan untuk mengatur nilai-nilai kustom.

```
aws greengrass create-function-definition-version \
--function-definition-id function-definition-id \
--functions '[
        {
            "FunctionArn": "arn:aws:lambda:::function:GGStreamManager:1",
            "FunctionConfiguration": {
                "Environment": {
                    "Variables": {
                        "STREAM_MANAGER_STORE_ROOT_DIR": "/data",
                        "STREAM_MANAGER_SERVER_PORT": "1234",
                        "STREAM_MANAGER_EXPORTER_THREAD_POOL_SIZE": "4"
                    }
                },
                "MemorySize":  4194304,
                "Pinned": true,
                "Timeout": 3
```

```
            },
            "Id": "streamManager"
        },
        {
            "FunctionArn": "arn:aws:lambda:us-
west-2:123456789012:function:MyLambdaFunction:MyAlias",
            "FunctionConfiguration": {
                "Executable": "myLambdaFunction.function_handler",
                "MemorySize": 16000,
                "Pinned": true,
                "Timeout": 5
            },
            "Id": "myLambdaFunction"
        },
        ... more user-defined functions
    ]
}'
```

> **Note**
>
> Untuk properti `FunctionConfiguration` ini, Anda mungkin mengetahui hal berikut:
>
> - `MemorySize` diatur ke 4194304 KB (4 GB) dengan pengaturan default. Anda selalu dapat mengubah nilai ini. Kami menganjurkan Anda untuk mengatur `MemorySize` ke setidaknya 128000 KB.
> - `Pinned` harus diatur ke `true`.
> - `Timeout` diperlukan oleh versi definisi fungsi, tetapi `GGStreamManager` tidak menggunakannya.

Nonaktifkan pengelola pengaliran

Contoh berikut menghilangkan fungsi `GGStreamManager` ini, yang menonaktifkan pengelola pengaliran.

```
aws greengrass create-function-definition-version \
--function-definition-id function-definition-id \
--functions '[
        {
```

```
            "FunctionArn": "arn:aws:lambda:us-
west-2:123456789012:function:MyLambdaFunction:MyAlias",
            "FunctionConfiguration": {
                "Executable": "myLambdaFunction.function_handler",
                "MemorySize": 16000,
                "Pinned": true,
                "Timeout": 5
            },
            "Id": "myLambdaFunction"
        },
        ... more user-defined functions
    ]
}'
```

> **Note**
>
> Jika Anda tidak ingin men-deploy fungsi Lambda, Anda dapat menghilangkan versi definisi fungsi sepenuhnya.

7. Salin `Arn` dari versi definisi fungsi dari output.
8. Buat versi grup yang berisi fungsi Lambda sistem.

   - Ganti *`id-grup`* dengan `Id` untuk grup.
   - Ganti*`core-definition-version-arn`*dengan`CoreDefinitionVersionArn`yang Anda salin dari versi grup terbaru.
   - Ganti*`function-definition-version-arn`*dengan`Arn`yang Anda salin untuk versi definisi fungsi baru.
   - Ganti ARN untuk komponen grup lainnya (sebagai contoh, `SubscriptionDefinitionVersionArn` atau `DeviceDefinitionVersionArn`) yang Anda salin dari versi grup terbaru.
   - Hapus parameter yang tidak terpakai. Sebagai contoh, Hapus `--resource-definition-version-arn` jika versi grup Anda tidak berisi sumber daya apa pun.

```
aws greengrass create-group-version \
--group-id group-id \
--core-definition-version-arn core-definition-version-arn \
--function-definition-version-arn function-definition-version-arn \
--device-definition-version-arn device-definition-version-arn \
```

```
--logger-definition-version-arn logger-definition-version-arn \
--resource-definition-version-arn resource-definition-version-arn \
--subscription-definition-version-arn subscription-definition-version-arn
```

9. Salin `Version` dari output. Ini adalah ID dari versi grup baru.
10. Men-deploy grup dengan versi grup baru.

    - Ganti *id-grup* dengan `Id` yang Anda salin untuk grup.
    - Ganti*group-version-id*dengan`Version`yang Anda salin untuk versi grup baru.

```
aws greengrass create-deployment \
--group-id group-id \
--group-version-id group-version-id \
--deployment-type NewDeployment
```

Ikuti prosedur ini jika Anda ingin mengedit pengaturan pengelola pengaliran lagi nanti. Pastikan untuk membuat versi definisi fungsi yang mencakup `GGStreamManager` fungsi dengan konfigurasi yang diperbarui. Versi grup harus mereferensi semua ARN versi komponen yang ingin Anda men-deploy ke core. Perubahan berlaku setelah grup di-deploy.

Lihat juga

- Mengelola aliran data
- the section called “Gunakan StreamManagerClient untuk bekerja dengan aliran”
- the section called “Konfigurasi ekspor untuk tujuan AWS Cloud yang didukung”
- the section called “Ekspor aliran data (konsol)”
- the section called “Ekspor aliran data (CLI)”

Gunakan StreamManagerClient untuk bekerja dengan aliran

Fungsi Lambda yang ditetapkan pengguna yang berjalan pada core AWS IoT Greengrass dapat menggunakan objek `StreamManagerClient` di dalam AWS IoT Greengrass Core SDK untuk membuat pengaliran di pengelola aliran lalu berinteraksi dengan pengaliran. Ketika fungsi Lambda

membuat pengaliran, ia menentukan AWS Cloud tujuan, prioritasisasi, dan kebijakan penyimpanan data serta ekspor lainnya untuk pengaliran tersebut. Untuk mengirim data ke pengelola pengaliran, fungsi Lambda menambahkan data ke pengaliran. Jika tujuan ekspor ditentukan untuk pengaliran, pengelola pengaliran mengekspor pengaliran secara otomatis.

> **Note**
>
> Biasanya, klien dari pengelola pengaliran adalah fungsi Lambda yang ditetapkan pengguna. Jika kasus bisnis Anda memerlukannya, Anda juga dapat mengizinkan proses non-Lambda berjalan pada core Greengrass (sebagai contoh, wadah Docker) untuk berinteraksi dengan pengelola pengaliran. Untuk informasi selengkapnya, lihat the section called “Autentikasi klien”.

Cuplikan dalam topik ini menunjukkan cara klien memanggil metode `StreamManagerClient` untuk bekerja dengan aliran. Untuk rincian implementasi tentang metode dan argumen mereka, gunakan tautan ke referensi SDK yang tercantum setelah setiap cuplikan. Untuk tutorial yang mencakup fungsi Python Lambda lengkap, lihat the section called “Ekspor aliran data (konsol)” atau the section called “Ekspor aliran data (CLI)”.

Fungsi Lambda Anda harus berjalan `StreamManagerClient` di luar dari fungsi handler. Jika berjalan di dalam handler, fungsi membuat `client` dan koneksi ke pengelola pengaliran setiap kali dipanggil.

> **Note**
>
> Jika Anda membuat contoh `StreamManagerClient` dalam handler, Anda harus secara tegas memanggil metode `close()` ketika `client` menyelesaikan pekerjaannya. Jika tidak, `client` akan membuat sambungan terbuka dan utas lain yang berjalan sampai skrip keluar.

`StreamManagerClient` mendukung operasi berikut:

- the section called “Membuat pengaliran pesan”
- the section called “Tambahkan pesan”
- the section called “Baca pesan”
- the section called “Daftar aliran”

- the section called “Jelaskan aliran pesan”
- the section called “Perbarui aliran pesan”
- the section called “Hapus aliran pesan”

Membuat pengaliran pesan

Untuk membuat pengaliran, fungsi Lambda yang ditetapkan pengguna memanggil metode pembuatan dan melewati di dalam `MessageStreamDefinition` objek. Objek ini menentukan nama unik untuk pengaliran dan menentukan cara pengelola pengaliran harus menangani data baru ketika ukuran pengaliran maksimum tercapai. Anda dapat menggunakan `MessageStreamDefinition` dan jenis datanya (seperti `ExportDefinition`, `StrategyOnFull`, dan `Persistence`) untuk menentukan properti pengaliran lainnya. Ini termasuk:

- Target AWS IoT Analytics, Kinesis Data Streams, AWS IoT SiteWise, dan tujuan Amazon S3 untuk ekspor otomatis. Untuk informasi selengkapnya, lihat the section called “Konfigurasi ekspor untuk tujuan AWS Cloud yang didukung”.
- Prioritas ekspor. Stream manager mengekspor aliran prioritas yang lebih tinggi sebelum aliran prioritas lebih rendah.
- Ukuran batch maksimum dan interval batch untuk AWS IoT Analytics, Kinesis Data Streams, dan tujuan AWS IoT SiteWise. Stream manager mengekspor pesan ketika salah satu kondisi terpenuhi.
- Waktu untuk tayang (TTL). Jumlah waktu untuk menjamin bahwa data aliran tersedia untuk diproses. Anda harus memastikan bahwa data dapat dikonsumsi dalam periode waktu ini. Ini bukan kebijakan penghapusan. Data mungkin tidak segera dihapus setelah periode TTL.
- Ketekunan aliran. Pilih untuk menyimpan aliran ke sistem file untuk mempertahankan data di seluruh restart inti atau menyimpan aliran dalam memori.
- Memulai nomor urut. Tentukan nomor urutan pesan yang akan digunakan sebagai pesan awal dalam ekspor.

Untuk informasi lebih lanjut tentang `MessageStreamDefinition`, lihat referensi SDK untuk bahasa target Anda:

- MessageStreamDefinition di SDK Java
- MessageStreamDefinition di SDK Node.js
- MessageStreamDefinition di SDK Python

> **Note**
>
> `StreamManagerClient` juga menyediakan target tujuan yang dapat Anda gunakan untuk mengekspor aliran ke server HTTP. Target ini ditujukan untuk tujuan pengujian saja. Hal ini tidak stabil atau didukung untuk digunakan di lingkungan produksi.

Setelah pengaliran dibuat, fungsi Lambda Anda dapat tambahkan pesan ke pengaliran untuk mengirim data untuk ekspor dan baca pesan dari pengaliran untuk pemrosesan lokal. Jumlah pengaliran yang Anda buat tergantung pada kemampuan perangkat keras dan kasus bisnis Anda. Salah satu strateginya adalah membuat aliran untuk setiap saluran target di AWS IoT Analytics atau aliran data Kinesis, meskipun Anda dapat menentukan beberapa target untuk suatu aliran. Aliran memiliki masa pakai yang tahan lama.

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- MinimumAWS IoT GreengrassVersi inti: 1.10.0
- MinimumAWS IoT GreengrassVersi SDK inti: Python: 1.5.0 | Java: 1.4.0 | Node.js: 1.6.0

> **Note**
>
> Membuat pengaliran dengan AWS IoT SiteWise atau tujuan ekspor Amazon S3 memiliki persyaratan sebagai berikut:
>
> - MinimumAWS IoT GreengrassVersi inti: 1.11.0
> - MinimumAWS IoT GreengrassVersi SDK inti: Python: 1.6.0 | Java: 1.5.0 | Node.js: 1.7.0

Contoh

Potongan berikut menciptakan aliran bernama `StreamName`. Potongan ini menentukan sifat aliran di `MessageStreamDefinition` dan jenis data bawahan.

Python

```
client = StreamManagerClient()
```

```
try:
    client.create_message_stream(MessageStreamDefinition(
        name="StreamName",  # Required.
        max_size=268435456,  # Default is 256 MB.
        stream_segment_size=16777216,  # Default is 16 MB.
        time_to_live_millis=None,  # By default, no TTL is enabled.
        strategy_on_full=StrategyOnFull.OverwriteOldestData,  # Required.
        persistence=Persistence.File,  # Default is File.
        flush_on_write=False,  # Default is false.
        export_definition=ExportDefinition(  # Optional. Choose where/how the stream
 is exported to the AWS Cloud.
            kinesis=None,
            iot_analytics=None,
            iot_sitewise=None,
            s3_task_executor=None
        )
    ))
except StreamManagerException:
    pass
    # Properly handle errors.
except ConnectionError or asyncio.TimeoutError:
    pass
    # Properly handle errors.
```

Referensi SDK Python: create_message_stream | MessageStreamDefinition

Java

```
try (final StreamManagerClient client =
 GreengrassClientBuilder.streamManagerClient().build()) {
    client.createMessageStream(
            new MessageStreamDefinition()
                    .withName("StreamName") // Required.
                    .withMaxSize(268435456L)  // Default is 256 MB.
                    .withStreamSegmentSize(16777216L)  // Default is 16 MB.
                    .withTimeToLiveMillis(null)  // By default, no TTL is enabled.
                    .withStrategyOnFull(StrategyOnFull.OverwriteOldestData)  //
 Required.
                    .withPersistence(Persistence.File)  // Default is File.
                    .withFlushOnWrite(false)  // Default is false.
                    .withExportDefinition(  // Optional. Choose where/how the stream
 is exported to the AWS Cloud.
                            new ExportDefinition()
```

```
                                    .withKinesis(null)
                                    .withIotAnalytics(null)
                                    .withIotSitewise(null)
                                    .withS3TaskExecutor(null)
                    )

    );
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Referensi SDK Java: createMessageStream | MessageStreamDefinition

Node.js

```
const client = new StreamManagerClient();
client.onConnected(async () => {
    try {
        await client.createMessageStream(
            new MessageStreamDefinition()
                .withName("StreamName") // Required.
                .withMaxSize(268435456)  // Default is 256 MB.
                .withStreamSegmentSize(16777216)  // Default is 16 MB.
                .withTimeToLiveMillis(null)  // By default, no TTL is enabled.
                .withStrategyOnFull(StrategyOnFull.OverwriteOldestData)  //
 Required.
                .withPersistence(Persistence.File)  // Default is File.
                .withFlushOnWrite(false)  // Default is false.
                .withExportDefinition(  // Optional. Choose where/how the stream is
 exported to the AWS Cloud.
                    new ExportDefinition()
                        .withKinesis(null)
                        .withIotAnalytics(null)
                        .withIotSitewise(null)
                        .withS3TaskExecutor(null)
                )
        );
    } catch (e) {
        // Properly handle errors.
    }
});
client.onError((err) => {
    // Properly handle connection errors.
    // This is called only when the connection to the StreamManager server fails.
```

```
});
```

Referensi SDK Node.js: createMessageStream | MessageStreamDefinition

Untuk informasi lebih lanjut tentang konfigurasi tujuan ekspor, lihatthe section called “Konfigurasi ekspor untuk tujuan AWS Cloud yang didukung”.

Tambahkan pesan

Untuk mengirim data ke pengelola pesan untuk ekspor, fungsi Lambda Anda menambahkan data ke pengaliran target. Tujuan ekspor menentukan tipe data yang lolos untuk metode ini.

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- MinimumAWS IoT GreengrassVersi inti: 1.10.0
- MinimumAWS IoT GreengrassVersi SDK inti: Python: 1.5.0 | Java: 1.4.0 | Node.js: 1.6.0

> **Note**
>
> Menambahkan pesan dengan AWS IoT SiteWise Atau tujuan ekspor Amazon S3 memiliki persyaratan sebagai berikut:
>
> - MinimumAWS IoT GreengrassVersi inti: 1.11.0
> - MinimumAWS IoT GreengrassVersi SDK inti: Python: 1.6.0 | Java: 1.5.0 | Node.js: 1.7.0

Contoh

AWS IoT Analytics atau tujuan ekspor Kinesis Data Streams

Cuplikan berikut menambahkan pesan ke pengaliran bernama `StreamName`. Untuk AWS IoT Analytics atau tujuan Kinesis Data Streams, fungsi Lambda Anda menambahkan setumpuk data.

Potongan ini memiliki persyaratan sebagai berikut:

- MinimumAWS IoT GreengrassVersi inti: 1.10.0

- MinimumAWS IoT GreengrassVersi SDK inti: Python: 1.5.0 | Java: 1.4.0 | Node.js: 1.6.0

Python

```
client = StreamManagerClient()

try:
    sequence_number = client.append_message(stream_name="StreamName",
 data=b'Arbitrary bytes data')
except StreamManagerException:
    pass
    # Properly handle errors.
except ConnectionError or asyncio.TimeoutError:
    pass
    # Properly handle errors.
```

Referensi SDK Python: append_message

Java

```
try (final StreamManagerClient client =
 GreengrassClientBuilder.streamManagerClient().build()) {
    long sequenceNumber = client.appendMessage("StreamName", "Arbitrary byte
 array".getBytes());
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Referensi SDK Java: appendMessage

Node.js

```
const client = new StreamManagerClient();
client.onConnected(async () => {
    try {
        const sequenceNumber = await client.appendMessage("StreamName",
 Buffer.from("Arbitrary byte array"));
    } catch (e) {
        // Properly handle errors.
    }
});
client.onError((err) => {
```

```
    // Properly handle connection errors.
    // This is called only when the connection to the StreamManager server fails.
});
```

Referensi SDK Node.js: appendMessage

Tujuan ekspor AWS IoT SiteWise

Cuplikan berikut menambahkan pesan ke pengaliran bernama `StreamName`. Untuk AWS IoT SiteWise tujuan, fungsi Lambda Anda menambahkan `PutAssetPropertyValueEntry` objek berseri. Untuk informasi selengkapnya, lihat the section called “Mengekspor ke AWS IoT SiteWise”.

> **Note**
>
> Saat Anda mengirim data ke AWS IoT SiteWise, data Anda harus memenuhi persyaratan tindakan `BatchPutAssetPropertyValue`. Untuk informasi selengkapnya, lihat:BatchPutAssetPropertyValue di Referensi API AWS IoT SiteWise.

Potongan ini memiliki persyaratan sebagai berikut:

- MinimumAWS IoT GreengrassVersi inti: 1.11.0
- MinimumAWS IoT GreengrassVersi SDK inti: Python: 1.6.0 | Java: 1.5.0 | Node.js: 1.7.0

Python

```
client = StreamManagerClient()

try:
    # SiteWise requires unique timestamps in all messages. Add some randomness to
 time and offset.

    # Note: To create a new asset property data, you should use the classes defined
 in the
    # greengrasssdk.stream_manager module.

    time_in_nanos = TimeInNanos(
        time_in_seconds=calendar.timegm(time.gmtime()) - random.randint(0, 60),
 offset_in_nanos=random.randint(0, 10000)
    )
```

```
    variant = Variant(double_value=random.random())
    asset = [AssetPropertyValue(value=variant, quality=Quality.GOOD,
 timestamp=time_in_nanos)]
    putAssetPropertyValueEntry =
 PutAssetPropertyValueEntry(entry_id=str(uuid.uuid4()),
 property_alias="PropertyAlias", property_values=asset)
    sequence_number = client.append_message(stream_name="StreamName",
 data=Util.validate_and_serialize_to_json_bytes(putAssetPropertyValueEntry))
except StreamManagerException:
    pass
    # Properly handle errors.
except ConnectionError or asyncio.TimeoutError:
    pass
    # Properly handle errors.
```

Referensi SDK Python: append_message | PutAssetPropertyValueEntry

Java

```
try (final StreamManagerClient client =
 GreengrassClientBuilder.streamManagerClient().build()) {
    Random rand = new Random();
    // Note: To create a new asset property data, you should use the classes defined
 in the
    // com.amazonaws.greengrass.streammanager.model.sitewise package.
    List<AssetPropertyValue> entries = new ArrayList<>() ;

    // IoTSiteWise requires unique timestamps in all messages. Add some randomness
 to time and offset.
    final int maxTimeRandomness = 60;
    final int maxOffsetRandomness = 10000;
    double randomValue = rand.nextDouble();
    TimeInNanos timestamp = new TimeInNanos()
            .withTimeInSeconds(Instant.now().getEpochSecond() -
 rand.nextInt(maxTimeRandomness))
            .withOffsetInNanos((long) (rand.nextInt(maxOffsetRandomness)));
    AssetPropertyValue entry = new AssetPropertyValue()
            .withValue(new Variant().withDoubleValue(randomValue))
            .withQuality(Quality.GOOD)
            .withTimestamp(timestamp);
    entries.add(entry);

    PutAssetPropertyValueEntry putAssetPropertyValueEntry = new
 PutAssetPropertyValueEntry()
```

```
            .withEntryId(UUID.randomUUID().toString())
            .withPropertyAlias("PropertyAlias")
            .withPropertyValues(entries);
    long sequenceNumber = client.appendMessage("StreamName",
 ValidateAndSerialize.validateAndSerializeToJsonBytes(putAssetPropertyValueEntry));
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Referensi SDK Java: appendMessage | PutAssetPropertyValueEntry

Node.js

```
const client = new StreamManagerClient();
client.onConnected(async () => {
    try {
        const maxTimeRandomness = 60;
        const maxOffsetRandomness = 10000;
        const randomValue = Math.random();
        // Note: To create a new asset property data, you should use the classes
 defined in the
        // aws-greengrass-core-sdk StreamManager module.
        const timestamp = new TimeInNanos()
            .withTimeInSeconds(Math.round(Date.now() / 1000) -
 Math.floor(Math.random() - maxTimeRandomness))
            .withOffsetInNanos(Math.floor(Math.random() * maxOffsetRandomness));
        const entry = new AssetPropertyValue()
            .withValue(new Variant().withDoubleValue(randomValue))
            .withQuality(Quality.GOOD)
            .withTimestamp(timestamp);

        const putAssetPropertyValueEntry =  new PutAssetPropertyValueEntry()
            .withEntryId(`${ENTRY_ID_PREFIX}${i}`)
            .withPropertyAlias("PropertyAlias")
            .withPropertyValues([entry]);
        const sequenceNumber = await client.appendMessage("StreamName",
 util.validateAndSerializeToJsonBytes(putAssetPropertyValueEntry));
    } catch (e) {
        // Properly handle errors.
    }
});
client.onError((err) => {
    // Properly handle connection errors.
    // This is called only when the connection to the StreamManager server fails.
```

```
});
```

Referensi SDK Node.js: appendMessage | PutAssetPropertyValueEntry

Tujuan ekspor Amazon S3

Cuplikan berikut menambahkan tugas ekspor ke pengaliran bernama `StreamName`. Untuk tujuan Amazon S3, fungsi Lambda Anda menambahkan `S3ExportTaskDefinition` objek berseri yang berisi informasi tentang file input sumber dan objek Amazon S3 target. Jika objek yang ditentukan tidak ada, Pengelola Pengaliran membuatnya untuk Anda. Untuk informasi selengkapnya, lihat the section called “Mengekspor ke Amazon S3”.

Cuplikan ini memiliki persyaratan sebagai berikut:

- MinimumAWS IoT GreengrassVersi inti: 1.11.0
- MinimumAWS IoT GreengrassVersi SDK inti: Python: 1.6.0 | Java: 1.5.0 | Node.js: 1.7.0

Python

```
client = StreamManagerClient()

try:
    # Append an Amazon S3 Task definition and print the sequence number.
    s3_export_task_definition = S3ExportTaskDefinition(input_url="URLToFile",
 bucket="BucketName", key="KeyName")
    sequence_number = client.append_message(stream_name="StreamName",
 data=Util.validate_and_serialize_to_json_bytes(s3_export_task_definition))
except StreamManagerException:
    pass
    # Properly handle errors.
except ConnectionError or asyncio.TimeoutError:
    pass
    # Properly handle errors.
```

Referensi SDK Python: append_message | S3ExportTaskDefinition

Java

```
try (final StreamManagerClient client =
 GreengrassClientBuilder.streamManagerClient().build()) {
```

```
    // Append an Amazon S3 export task definition and print the sequence number.
    S3ExportTaskDefinition s3ExportTaskDefinition = new S3ExportTaskDefinition()
        .withBucket("BucketName")
        .withKey("KeyName")
        .withInputUrl("URLToFile");
    long sequenceNumber = client.appendMessage("StreamName",
 ValidateAndSerialize.validateAndSerializeToJsonBytes(s3ExportTaskDefinition));
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Referensi SDK Java: appendMessage | S3ExportTaskDefinition

Node.js

```
const client = new StreamManagerClient();
client.onConnected(async () => {
    try {
     // Append an Amazon S3 export task definition and print the sequence number.
     const taskDefinition = new S3ExportTaskDefinition()
        .withBucket("BucketName")
        .withKey("KeyName")
        .withInputUrl("URLToFile");
        const sequenceNumber = await client.appendMessage("StreamName",
 util.validateAndSerializeToJsonBytes(taskDefinition)));
    } catch (e) {
        // Properly handle errors.
    }
});
client.onError((err) => {
    // Properly handle connection errors.
    // This is called only when the connection to the StreamManager server fails.
});
```

Referensi SDK Node.js: appendMessage | S3ExportTaskDefinition

Baca pesan

Baca pesan dari suatu aliran.

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- MinimumAWS IoT GreengrassVersi inti: 1.10.0
- MinimumAWS IoT GreengrassVersi SDK inti: Python: 1.5.0 | Java: 1.4.0 | Node.js: 1.6.0

Contoh

Potongan berikut membaca pesan dari aliran bernama `StreamName`. Metode membaca mengambil objek `ReadMessagesOptions` opsional yang menentukan urutan nomor untuk memulai membaca, jumlah minimum dan maksimum yang dibaca, dan batas waktu untuk membaca pesan.

Python

```
client = StreamManagerClient()

try:
    message_list = client.read_messages(
        stream_name="StreamName",
        # By default, if no options are specified, it tries to read one message from
 the beginning of the stream.
        options=ReadMessagesOptions(
            desired_start_sequence_number=100,
            # Try to read from sequence number 100 or greater. By default, this is
 0.
            min_message_count=10,
            # Try to read 10 messages. If 10 messages are not available, then
 NotEnoughMessagesException is raised. By default, this is 1.
            max_message_count=100,  # Accept up to 100 messages. By default this is
 1.
            read_timeout_millis=5000
            # Try to wait at most 5 seconds for the min_messsage_count to be
 fulfilled. By default, this is 0, which immediately returns the messages or an
 exception.
        )
    )
except StreamManagerException:
    pass
    # Properly handle errors.
except ConnectionError or asyncio.TimeoutError:
    pass
```

```
    # Properly handle errors.
```

Referensi SDK Python: read_messages | ReadMessagesOptions

Java

```
try (final StreamManagerClient client =
 GreengrassClientBuilder.streamManagerClient().build()) {
    List<Message> messages = client.readMessages("StreamName",
            // By default, if no options are specified, it tries to read one message
 from the beginning of the stream.
            new ReadMessagesOptions()
                    // Try to read from sequence number 100 or greater. By default
 this is 0.
                    .withDesiredStartSequenceNumber(100L)
                    // Try to read 10 messages. If 10 messages are not available,
 then NotEnoughMessagesException is raised. By default, this is 1.
                    .withMinMessageCount(10L)
                    // Accept up to 100 messages. By default this is 1.
                    .withMaxMessageCount(100L)
                    // Try to wait at most 5 seconds for the min_messsage_count to
 be fulfilled. By default, this is 0, which immediately returns the messages or an
 exception.
                    .withReadTimeoutMillis(Duration.ofSeconds(5L).toMillis())
    );
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Referensi SDK Java: readMessages | ReadMessagesOptions

Node.js

```
const client = new StreamManagerClient();
client.onConnected(async () => {
    try {
        const messages = await client.readMessages("StreamName",
            // By default, if no options are specified, it tries to read one message
 from the beginning of the stream.
            new ReadMessagesOptions()
                // Try to read from sequence number 100 or greater. By default this
 is 0.
                .withDesiredStartSequenceNumber(100)
```

```
                // Try to read 10 messages. If 10 messages are not available, then
 NotEnoughMessagesException is thrown. By default, this is 1.
                .withMinMessageCount(10)
                // Accept up to 100 messages. By default this is 1.
                .withMaxMessageCount(100)
                // Try to wait at most 5 seconds for the minMessageCount to be
 fulfilled. By default, this is 0, which immediately returns the messages or an
 exception.
                .withReadTimeoutMillis(5 * 1000)
        );
    } catch (e) {
        // Properly handle errors.
    }
});
client.onError((err) => {
    // Properly handle connection errors.
    // This is called only when the connection to the StreamManager server fails.
});
```

Referensi SDK Java: readMessages | ReadMessagesOptions

Daftar aliran

Dapatkan daftar aliran di stream manager.

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- MinimumAWS IoT GreengrassVersi inti: 1.10.0
- MinimumAWS IoT GreengrassVersi SDK inti: Python: 1.5.0 | Java: 1.4.0 | Node.js: 1.6.0

Contoh

Potongan berikut mendapat daftar aliran (dengan nama) di stream manager.

Python

```
client = StreamManagerClient()
```

```
try:
    stream_names = client.list_streams()
except StreamManagerException:
    pass
    # Properly handle errors.
except ConnectionError or asyncio.TimeoutError:
    pass
    # Properly handle errors.
```

Referensi SDK Python: list_streams

Java

```
try (final StreamManagerClient client =
 GreengrassClientBuilder.streamManagerClient().build()) {
    List<String> streamNames = client.listStreams();
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Referensi SDK Java: listStreams

Node.js

```
const client = new StreamManagerClient();
client.onConnected(async () => {
    try {
        const streams = await client.listStreams();
    } catch (e) {
        // Properly handle errors.
    }
});
client.onError((err) => {
    // Properly handle connection errors.
    // This is called only when the connection to the StreamManager server fails.
});
```

Referensi SDK Node.js: listStreams

Jelaskan aliran pesan

Dapatkan metadata tentang aliran, termasuk definisi, ukuran, dan status ekspor aliran.

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- MinimumAWS IoT GreengrassVersi inti: 1.10.0
- MinimumAWS IoT GreengrassVersi SDK inti: Python: 1.5.0 | Java: 1.4.0 | Node.js: 1.6.0

Contoh

Potongan berikut mendapat metadata tentang aliran bernama `StreamName`, termasuk status definisi, ukuran, dan pengekspor aliran.

Python

```
client = StreamManagerClient()

try:
    stream_description = client.describe_message_stream(stream_name="StreamName")
    if stream_description.export_statuses[0].error_message:
        # The last export of export destination 0 failed with some error
        # Here is the last sequence number that was successfully exported
        stream_description.export_statuses[0].last_exported_sequence_number

    if (stream_description.storage_status.newest_sequence_number >
            stream_description.export_statuses[0].last_exported_sequence_number):
        pass
        # The end of the stream is ahead of the last exported sequence number
except StreamManagerException:
    pass
    # Properly handle errors.
except ConnectionError or asyncio.TimeoutError:
    pass
    # Properly handle errors.
```

Referensi SDK Python: describe_message_stream

Java

```
try (final StreamManagerClient client =
 GreengrassClientBuilder.streamManagerClient().build()) {
    MessageStreamInfo description = client.describeMessageStream("StreamName");
    String lastErrorMessage =
 description.getExportStatuses().get(0).getErrorMessage();
    if (lastErrorMessage != null && !lastErrorMessage.equals("")) {
        // The last export of export destination 0 failed with some error.
        // Here is the last sequence number that was successfully exported.
        description.getExportStatuses().get(0).getLastExportedSequenceNumber();
    }

    if (description.getStorageStatus().getNewestSequenceNumber() >
            description.getExportStatuses().get(0).getLastExportedSequenceNumber())
 {
        // The end of the stream is ahead of the last exported sequence number.
    }
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Referensi SDK Java: describeMessageStream

Node.js

```
const client = new StreamManagerClient();
client.onConnected(async () => {
    try {
        const description = await client.describeMessageStream("StreamName");
        const lastErrorMessage = description.exportStatuses[0].errorMessage;
        if (lastErrorMessage) {
            // The last export of export destination 0 failed with some error.
            // Here is the last sequence number that was successfully exported.
            description.exportStatuses[0].lastExportedSequenceNumber;
        }

        if (description.storageStatus.newestSequenceNumber >
            description.exportStatuses[0].lastExportedSequenceNumber) {
            // The end of the stream is ahead of the last exported sequence number.
        }
    } catch (e) {
        // Properly handle errors.
    }
```

```
});
client.onError((err) => {
    // Properly handle connection errors.
    // This is called only when the connection to the StreamManager server fails.
});
```

Referensi SDK Node.js: describeMessageStream

Perbarui aliran pesan

Perbarui properti dari aliran yang ada. Anda mungkin ingin memperbarui aliran jika kebutuhan Anda berubah setelah aliran dibuat. Misalnya:

- Tambahkan konfigurasi ekspor baru untuk tujuan AWS Cloud.
- Tingkatkan ukuran maksimum aliran untuk mengubah cara data diekspor atau disimpan. Misalnya, ukuran aliran yang dikombinasikan dengan strategi Anda pada pengaturan penuh dapat mengakibatkan data dihapus atau ditolak sebelum stream manager dapat memprosesnya.
- Jeda dan lanjutkan ekspor; sebagai contoh, jika tugas ekspor berjalan lama dan Anda ingin menjatah data unggahan Anda.

Fungsi Lambda Anda mengikuti proses tingkat tinggi ini untuk memperbarui pengaliran:

1. Dapatkan deskripsi aliran.
2. Perbarui properti target pada `MessageStreamDefinition` dan objek bawahan yang sesuai.
3. Lewati `MessageStreamDefinition` yang diperbarui. Pastikan untuk menyertakan definisi objek lengkap untuk aliran yang diperbarui. Properti yang tidak terdefinisi kembali ke nilai default.

   Anda dapat menentukan nomor urutan pesan yang akan digunakan sebagai pesan awal dalam ekspor.

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- MinimumAWS IoT GreengrassVersi inti: 1.11.0
- MinimumAWS IoT GreengrassVersi SDK inti: Python: 1.6.0 | Java: 1.5.0 | Node.js: 1.7.0

Contoh

Potongan berikut menciptakan aliran bernama `StreamName`. Potongan ini memperbarui beberapa properti aliran yang mengekspor ke Kinesis Data Streams.

Python

```
client = StreamManagerClient()

try:
    message_stream_info = client.describe_message_stream(STREAM_NAME)
    message_stream_info.definition.max_size=536870912
    message_stream_info.definition.stream_segment_size=33554432
    message_stream_info.definition.time_to_live_millis=3600000
    message_stream_info.definition.strategy_on_full=StrategyOnFull.RejectNewData
    message_stream_info.definition.persistence=Persistence.Memory
    message_stream_info.definition.flush_on_write=False
    message_stream_info.definition.export_definition.kinesis=
        [KinesisConfig(
            # Updating Export definition to add a Kinesis Stream configuration.
            identifier=str(uuid.uuid4()), kinesis_stream_name=str(uuid.uuid4()))]
    client.update_message_stream(message_stream_info.definition)
except StreamManagerException:
    pass
    # Properly handle errors.
except ConnectionError or asyncio.TimeoutError:
    pass
    # Properly handle errors.
```

Referensi SDK Python: updateMessageStream | MessageStreamDefinition

Java

```
try (final StreamManagerClient client =
 GreengrassClientBuilder.streamManagerClient().build()) {
    MessageStreamInfo messageStreamInfo = client.describeMessageStream(STREAM_NAME);
    // Update the message stream with new values.
    client.updateMessageStream(
        messageStreamInfo.getDefinition()
            .withStrategyOnFull(StrategyOnFull.RejectNewData) // Required. Updating
 Strategy on full to reject new data.
            // Max Size update should be greater than initial Max Size defined in
 Create Message Stream request
```

```
            .withMaxSize(536870912L) // Update Max Size to 512 MB.
            .withStreamSegmentSize(33554432L) // Update Segment Size to 32 MB.
            .withFlushOnWrite(true) // Update flush on write to true.
            .withPersistence(Persistence.Memory) // Update the persistence to
 Memory.
            .withTimeToLiveMillis(3600000L)  // Update TTL to 1 hour.
            .withExportDefinition(
                // Optional. Choose where/how the stream is exported to the AWS
 Cloud.
                messageStreamInfo.getDefinition().getExportDefinition().
                    // Updating Export definition to add a Kinesis Stream
 configuration.
                    .withKinesis(new ArrayList<KinesisConfig>() {{
                        add(new KinesisConfig()
                            .withIdentifier(EXPORT_IDENTIFIER)
                            .withKinesisStreamName("test"));
                        }})
            );
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Referensi SDK Java: update_message_stream | MessageStreamDefinition

Node.js

```
const client = new StreamManagerClient();
client.onConnected(async () => {
    try {
        const messageStreamInfo = await c.describeMessageStream(STREAM_NAME);
        await client.updateMessageStream(
            messageStreamInfo.definition
                // Max Size update should be greater than initial Max Size defined
 in Create Message Stream request
                .withMaxSize(536870912)  // Default is 256 MB. Updating Max Size to
 512 MB.
                .withStreamSegmentSize(33554432)  // Default is 16 MB. Updating
 Segment Size to 32 MB.
                .withTimeToLiveMillis(3600000)  // By default, no TTL is enabled.
 Update TTL to 1 hour.
                .withStrategyOnFull(StrategyOnFull.RejectNewData)  // Required.
 Updating Strategy on full to reject new data.
                .withPersistence(Persistence.Memory)  // Default is File. Update the
 persistence to Memory
```

```
                .withFlushOnWrite(true)  // Default is false. Updating to true.
                .withExportDefinition(
                    // Optional. Choose where/how the stream is exported to the AWS
 Cloud.
                    messageStreamInfo.definition.exportDefinition
                        // Updating Export definition to add a Kinesis Stream
 configuration.
                        .withKinesis([new
 KinesisConfig().withIdentifier(uuidv4()).withKinesisStreamName(uuidv4())])
                )
        );
    } catch (e) {
        // Properly handle errors.
    }
});
client.onError((err) => {
    // Properly handle connection errors.
    // This is called only when the connection to the StreamManager server fails.
});
```

Referensi SDK Node.js: updateMessageStream | MessageStreamDefinition

Kendala untuk memperbarui aliran

Kendala berikut berlaku saat memperbarui aliran. Kecuali tercantum dalam daftar berikut, pembaruan berlaku segera.

- Anda tidak dapat memperbarui kegigihan aliran. Untuk mengubah perilaku ini, hapus aliran dan buat Stream yang menentukan kebijakan kegigihan baru.
- Anda dapat memperbarui ukuran maksimum aliran hanya dalam kondisi berikut:
  - Ukuran maksimum harus lebih besar atau sama dengan ukuran aliran. Untuk menemukan informasi ini,menggambarkan aliran dan kemudian periksa status penyimpanan yang dikembalikan`MessageStreamInfo`objek.
  - Ukuran maksimum harus lebih besar atau sama dengan ukuran segmen aliran.
- Anda dapat memperbarui ukuran segmen aliran ke nilai kurang dari ukuran maksimum aliran tersebut. Pengaturan yang diperbarui berlaku untuk segmen baru.
- Pembaruan ke properti waktu untuk tayang (TTL) berlaku untuk operasi append yang baru. Jika Anda mengurangi nilai ini, stream manager juga dapat menghapus segmen yang ada yang melebihi TTL.

- Pembaruan untuk strategi tersebut pada properti penuh berlaku untuk operasi append yang baru. Jika Anda menetapkan strategi untuk menimpa data tertua, stream manager juga dapat menimpa segmen yang ada berdasarkan pengaturan yang baru.
- Pembaruan untuk properti flush on write berlaku untuk pesan baru.
- Pembaruan untuk konfigurasi ekspor berlaku untuk ekspor baru. Permintaan pembaruan harus mencakup semua konfigurasi ekspor yang ingin Anda dukung. Jika tidak, stream manager akan menghapusnya.
  - Saat Anda memperbarui konfigurasi ekspor, tentukan pengenal konfigurasi ekspor target.
  - Untuk menambahkan konfigurasi ekspor, tentukan pengenal unik untuk konfigurasi ekspor baru.
  - Untuk menghapus konfigurasi ekspor, hapus konfigurasi ekspor.
- Untuk memperbarui nomor urutan awal konfigurasi ekspor di suatu aliran, Anda harus menentukan nilai yang kurang dari nomor urut terbaru. Untuk menemukan informasi ini,menggambarkan aliran dan kemudian periksa status penyimpanan yang dikembalikan`MessageStreamInfo`objek.

Hapus aliran pesan

Hapus aliran. Bila Anda menghapus suatu aliran, semua data yang tersimpan untuk aliran itu akan dihapus dari disk.

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- MinimumAWS IoT GreengrassVersi inti: 1.10.0
- MinimumAWS IoT GreengrassVersi SDK inti: Python: 1.5.0 | Java: 1.4.0 | Node.js: 1.6.0

Contoh

Potongan berikut menghapus aliran bernama `StreamName`.

Python

```
client = StreamManagerClient()
```

```
try:
    client.delete_message_stream(stream_name="StreamName")
except StreamManagerException:
    pass
    # Properly handle errors.
except ConnectionError or asyncio.TimeoutError:
    pass
    # Properly handle errors.
```

Referensi SDK Python: deleteMessageStream

Java

```
try (final StreamManagerClient client =
 GreengrassClientBuilder.streamManagerClient().build()) {
    client.deleteMessageStream("StreamName");
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Referensi SDK Java: delete_message_stream

Node.js

```
const client = new StreamManagerClient();
client.onConnected(async () => {
    try {
        await client.deleteMessageStream("StreamName");
    } catch (e) {
        // Properly handle errors.
    }
});
client.onError((err) => {
    // Properly handle connection errors.
    // This is called only when the connection to the StreamManager server fails.
});
```

Referensi SDK Node.js: deleteMessageStream

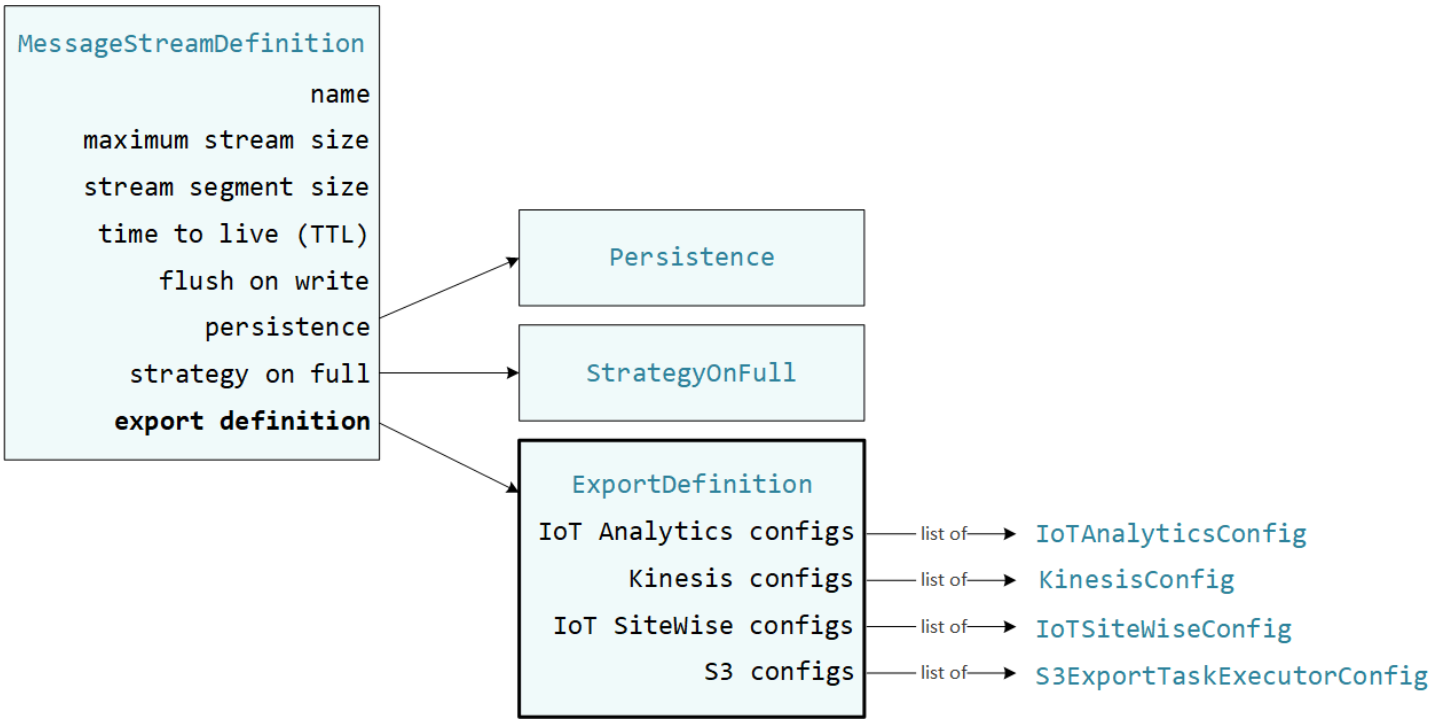
Lihat juga

- Mengelola aliran data

- the section called “Konfigurasi pengelola pengaliran”
- the section called “Konfigurasi ekspor untuk tujuan AWS Cloud yang didukung”
- the section called “Ekspor aliran data (konsol)”
- the section called “Ekspor aliran data (CLI)”
- `StreamManagerClient` di AWS IoT Greengrass Referensi SDK core:
  - Python
  - Java
  - Node.js

Konfigurasi ekspor untuk tujuan AWS Cloud yang didukung

Gunakan fungsi Lambda yang ditentukan pengguna `StreamManagerClient` di dalam AWS IoT Greengrass SDK core untuk berinteraksi dengan pengelola pengaliran. Ketika fungsi Lambda membuat pengaliran atau memperbarui pengaliran, fungsi ini melewati objek `MessageStreamDefinition` yang mewakili properti pengaliran, termasuk definisi ekspor. Objek `ExportDefinition` berisi konfigurasi ekspor yang ditentukan untuk pengaliran. Stream manager menggunakan konfigurasi ekspor ini untuk menentukan di mana dan bagaimana untuk mengekspor aliran tersebut.



Anda dapat menentukan konfigurasi ekspor nol atau lebih pada suatu aliran, termasuk beberapa konfigurasi ekspor untuk satu jenis tujuan. Misalnya, Anda dapat mengekspor aliran ke dua saluran AWS IoT Analytics dan satu Kinesis data stream.

Untuk usaha ekspor yang gagal, stream manager terus-menerus mencoba mengekspor data ke AWS Cloud pada interval hingga lima menit. Jumlah upaya coba lagi tidak memiliki batas maksimum.

> **Note**
>
> `StreamManagerClient` juga menyediakan target tujuan yang dapat Anda gunakan untuk mengekspor aliran ke server HTTP. Target ini ditujukan untuk tujuan pengujian saja. Target ini tidak stabil atau didukung untuk digunakan di lingkungan produksi.

Tujuan AWS Cloud yang didukung

- AWS IoT Analytics saluran
- Amazon Kinesis data streams
- Properti aset AWS IoT SiteWise
- Objek Amazon S3

Anda bertanggung jawab untuk mempertahankan AWS Cloud sumber daya ini.

AWS IoT Analytics saluran

Stream manager mendukung ekspor otomatis ke AWS IoT Analytics. AWS IoT Analytics memungkinkan Anda melakukan analisis lanjutan pada data Anda untuk membantu membuat keputusan bisnis dan meningkatkan model machine learning. Untuk informasi lebih lanjut, lihat Apa AWS IoT Analytics? dalam AWS IoT Analytics Panduan Pengguna.

Dalam SDK Core AWS IoT Greengrass tersebut, fungsi Lambda Anda menggunakan `IoTAnalyticsConfig` untuk menentukan konfigurasi ekspor untuk jenis tujuan ini. Untuk informasi lebih lanjut, lihat referensi SDK untuk bahasa target Anda:

- IoTAnalyticsConfig di SDK Python
- IoTAnalyticsConfig di SDK Java
- IoTAnalyticsConfig di SDK Node.js

Persyaratan

Tujuan ekspor ini memiliki persyaratan sebagai berikut:

- Saluran target di dalam AWS IoT Analytics harus ada di dalam Akun AWS yang sama dan Wilayah AWS sebagai grup Greengrass.
- Sebuah the section called “Peran grup Greengrass” harus memberikan `iotanalytics:BatchPutMessage` izin untuk menargetkan saluran. Misalnya:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iotanalytics:BatchPutMessage"
            ],
            "Resource": [
                "arn:aws:iotanalytics:region:account-id:channel/channel_1_name",
                "arn:aws:iotanalytics:region:account-id:channel/channel_2_name"
            ]
        }
    ]
}
```

  Anda dapat memberikan akses terperinci atau bersyarat ke sumber daya, misalnya dengan menggunakan skema penamaan wildcard `*`. Untuk informasi lebih lanjut, lihat Menambahkan dan menghapus kebijakan IAM dalam Panduan Pengguna IAM.

Mengekspor ke AWS IoT Analytics

Untuk membuat aliran yang mengekspor ke AWS IoT Analytics, fungsi Lambda Anda buat aliran dengan definisi ekspor yang mencakup satu atau lebih `IoTAnalyticsConfig` objek. Objek ini menentukan pengaturan ekspor, seperti saluran target, ukuran batch, batch interval, dan prioritas.

Ketika fungsi Lambda Anda menerima data dari perangkat, mereka menambahkan pesan yang berisi gumpalan data ke aliran target.

Kemudian, manajer pengaliran mengekspor data berdasarkan pengaturan batch dan prioritas yang ditentukan di dalam konfigurasi ekspor pengaliran.

Amazon Kinesis data streams

Pengelola pengaliran mendukung ekspor otomatis ke Amazon Kinesis Data Streams. Kinesis Data Streams umumnya digunakan untuk mengumpulkan data volume tinggi dan memuatnya ke dalam gudang data atau klaster map-reduce. Untuk informasi lebih lanjut, lihat Apa itu Amazon Kinesis Data Streams? dalam Panduan Developer Amazon Kinesis.

Dalam SDK Core AWS IoT Greengrass tersebut, fungsi Lambda Anda menggunakan `KinesisConfig` untuk menentukan konfigurasi ekspor untuk jenis tujuan ini. Untuk informasi lebih lanjut, lihat referensi SDK untuk bahasa target Anda:

- KinesisConfig di SDK Python
- KinesisConfig di SDK Java
- KinesisConfig di SDK Node.js

Persyaratan

Tujuan ekspor ini memiliki persyaratan sebagai berikut:

- Pengaliran target di dalam Kinesis Data Streams harus sama Akun AWS dan Wilayah AWS sebagai grup Greengrass.
- Sebuah the section called “Peran grup Greengrass” harus memberikan `kinesis:PutRecords` izin untuk menargetkan pengaliran data. Misalnya:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:PutRecords"
            ],
            "Resource": [
                "arn:aws:kinesis:region:account-id:stream/stream_1_name",
                "arn:aws:kinesis:region:account-id:stream/stream_2_name"
            ]
        }
    ]
```

```
}
```

Anda dapat memberikan akses terperinci atau bersyarat ke sumber daya, misalnya dengan menggunakan skema penamaan wildcard `*`. Untuk informasi lebih lanjut, lihat Menambahkan dan menghapus kebijakan IAM dalam Panduan Pengguna IAM.

Mengekspor ke Kinesis Data Streams

Untuk membuat pengaliran yang mengeksport ke Kinesis Data Streams, fungsi Lambda anda buat pengaliran dengan definisi ekspor yang mencakup satu atau lebih `KinesisConfig` objek. Objek ini menentukan pengaturan ekspor, seperti aliran data target, ukuran batch, interval batch, dan prioritas.

Ketika fungsi Lambda Anda menerima data dari perangkat, mereka menambahkan pesan yang berisi gumpalan data ke aliran target. Kemudian, manajer pengaliran mengekspor data berdasarkan pengaturan batch dan prioritas yang ditentukan di dalam konfigurasi ekspor pengaliran.

Stream manager menghasilkan UUID acak yang unik sebagai kunci partisi untuk setiap rekaman yang diunggah ke Amazon Kinesis.

Properti aset AWS IoT SiteWise

Stream manager mendukung ekspor otomatis ke AWS IoT SiteWise. AWS IoT SiteWise memungkinkan Anda mengumpulkan, mengatur, dan menganalisis data dari peralatan industri dalam skala besar. Untuk informasi lebih lanjut, lihat Apa AWS IoT SiteWise? dalam AWS IoT SiteWise Panduan Pengguna.

Dalam SDK Core AWS IoT Greengrass tersebut, fungsi Lambda Anda menggunakan `IoTSiteWiseConfig` untuk menentukan konfigurasi ekspor untuk jenis tujuan ini. Untuk informasi lebih lanjut, lihat referensi SDK untuk bahasa target Anda:

- IotSiteWiseConfig di SDK Python
- IotSiteWiseConfig di SDK Java
- IotSiteWiseconfig dalam Node.js SDK

> **Note**
>
> AWS juga menyediakan the section called “IoT SiteWise”, yang merupakan solusi pra-dibuat yang dapat Anda gunakan dengan sumber OPC-UA.

Persyaratan

Tujuan ekspor ini memiliki persyaratan sebagai berikut:

- Targetkan properti aset di AWS IoT SiteWise harus dalam yang sama Akun AWS dan Wilayah AWS sebagai grup Greengrass.

  > **Note**
  >
  > Untuk daftar Wilayah yang didukung AWS IoT SiteWise tersebut, lihat AWS IoT SiteWise kuota dan titik akhir dalam AWS Referensi Umum.

- Sebuah the section called “Peran grup Greengrass” harus mengizinkan `iotsitewise:BatchPutAssetPropertyValue` izin untuk menargetkan properti aset. Kebijakan berikut menggunakan kunci kondisi `iotsitewise:assetHierarchyPath` untuk memberikan akses ke aset akar target dan anak-anaknya. Anda dapat menghapus `Condition` dari kebijakan untuk mengizinkan akses ke semua aset AWS IoT SiteWise atau menentukan ARN aset individu.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
             "Effect": "Allow",
             "Action": "iotsitewise:BatchPutAssetPropertyValue",
             "Resource": "*",
             "Condition": {
                 "StringLike": {
                     "iotsitewise:assetHierarchyPath": [
                         "/root node asset ID",
                         "/root node asset ID/*"
                     ]
                 }
             }
```

```
        }
    ]
}
```

Anda dapat memberikan akses terperinci atau bersyarat ke sumber daya, misalnya dengan menggunakan skema penamaan wildcard *. Untuk informasi lebih lanjut, lihat Menambahkan dan menghapus kebijakan IAM dalam Panduan Pengguna IAM.

Untuk informasi keamanan penting, lihat BatchPutAssetPropertyValue otorisasidiAWS IoT SiteWisePanduan Pengguna.

Mengekspor ke AWS IoT SiteWise

Untuk membuat aliran yang mengekspor ke AWS IoT SiteWise, fungsi Lambda Anda buat aliran dengan definisi ekspor yang mencakup satu atau lebih `IoTSiteWiseConfig` objek. Objek ini mendefinisikan pengaturan ekspor, seperti ukuran batch, batch interval, dan prioritas.

Ketika fungsi Lambda Anda menerima data properti aset dari perangkat, mereka menambahkan pesan yang berisi data ke pengaliran target. Pesan adalah objek `PutAssetPropertyValueEntry` serial JSON yang berisi nilai properti untuk satu atau lebih properti aset. Untuk informasi selengkapnya, lihat: Tambahkan pesan untuk tujuan ekspor AWS IoT SiteWise.

> **Note**
>
> Saat Anda mengirim data ke AWS IoT SiteWise, data Anda harus memenuhi persyaratan tindakan `BatchPutAssetPropertyValue`. Untuk informasi selengkapnya, lihat BatchPutAssetPropertyValue di Referensi API AWS IoT SiteWise.

Kemudian, manajer pengaliran mengekspor data berdasarkan pengaturan batch dan prioritas yang ditentukan di dalam konfigurasi ekspor pengaliran.

Anda dapat menyesuaikan pengaturan pengelola pengaliran dan logika fungsi Lambda untuk merancang strategi ekspor Anda. Misalnya:

- Untuk ekspor yang mendekati real time, atur ukuran batch dan pengaturan interval yang rendah dan tambahkan data ke aliran saat diterima.

- Untuk mengoptimalkan batching, mengurangi kendala bandwidth, atau meminimalkan biaya, fungsi Lambda Anda dapat mengumpulkan timestamp-quality-value Titik data (TQV) diterima untuk properti aset tunggal sebelum menambahkan data ke aliran. Salah satu strateginya adalah mengelompokkan entri hingga 10 kombinasi aset properti yang berbeda, atau alias properti, dalam satu pesan, dan bukan mengirim lebih dari satu entri untuk properti yang sama. Hal ini membantu manajer pengaliran untuk tetap berada dalam kuota AWS IoT SiteWise.

Objek Amazon S3

Stream manager mendukung ekspor otomatis ke Amazon S3. Sebagai contoh, Anda dapat menggunakan Amazon S3 untuk menyimpan dan mengambil sejumlah besar data. Untuk informasi lebih lanjut, lihat Apa Amazon S3? dalam Panduan Developer Amazon Simple Storage Service.

Dalam SDK Core AWS IoT Greengrass tersebut, fungsi Lambda Anda menggunakan `S3ExportTaskExecutorConfig` untuk menentukan konfigurasi ekspor untuk jenis tujuan ini. Untuk informasi lebih lanjut, lihat referensi SDK untuk bahasa target Anda:

- S3ExportTaskExecutorConfig dalam SDK Python
- S3ExportTaskExecutorConfig dalam SDK Java
- S3ExportTaskExecutorConfig dalam SDK Node.js

Persyaratan

Tujuan ekspor ini memiliki persyaratan sebagai berikut:

- Target Amazon S3 bucket harus berada di Akun AWS sebagai grup Greengrass.
- Jika kontainerisasi default untuk grup Greengrass adalah kontainer Greengrass, Anda harus mengatur parameter STREAM_MANAGER_READ_ONLY_DIRS untuk menggunakan direktori file input yang berada di bawah `/tmp` atau tidak pada sistem file root.
- Jika fungsi Lambda berjalan di kontainer Greengrass mode menulis file input ke direktori file input, Anda harus membuat sumber daya volume lokal untuk direktori dan mount direktori ke kontainer dengan izin menulis. Hal ini memastikan bahwa file ditulis ke sistem file root dan terlihat di luar kontainer. Untuk informasi selengkapnya, lihat Akses sumber daya lokal.
- Sebuah the section called “Peran grup Greengrass” harus mengizinkan izin berikut untuk target bucket. Misalnya:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:AbortMultipartUpload",
                "s3:ListMultipartUploadParts"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-1-name/*",
                "arn:aws:s3:::bucket-2-name/*"
            ]
        }
    ]
}
```

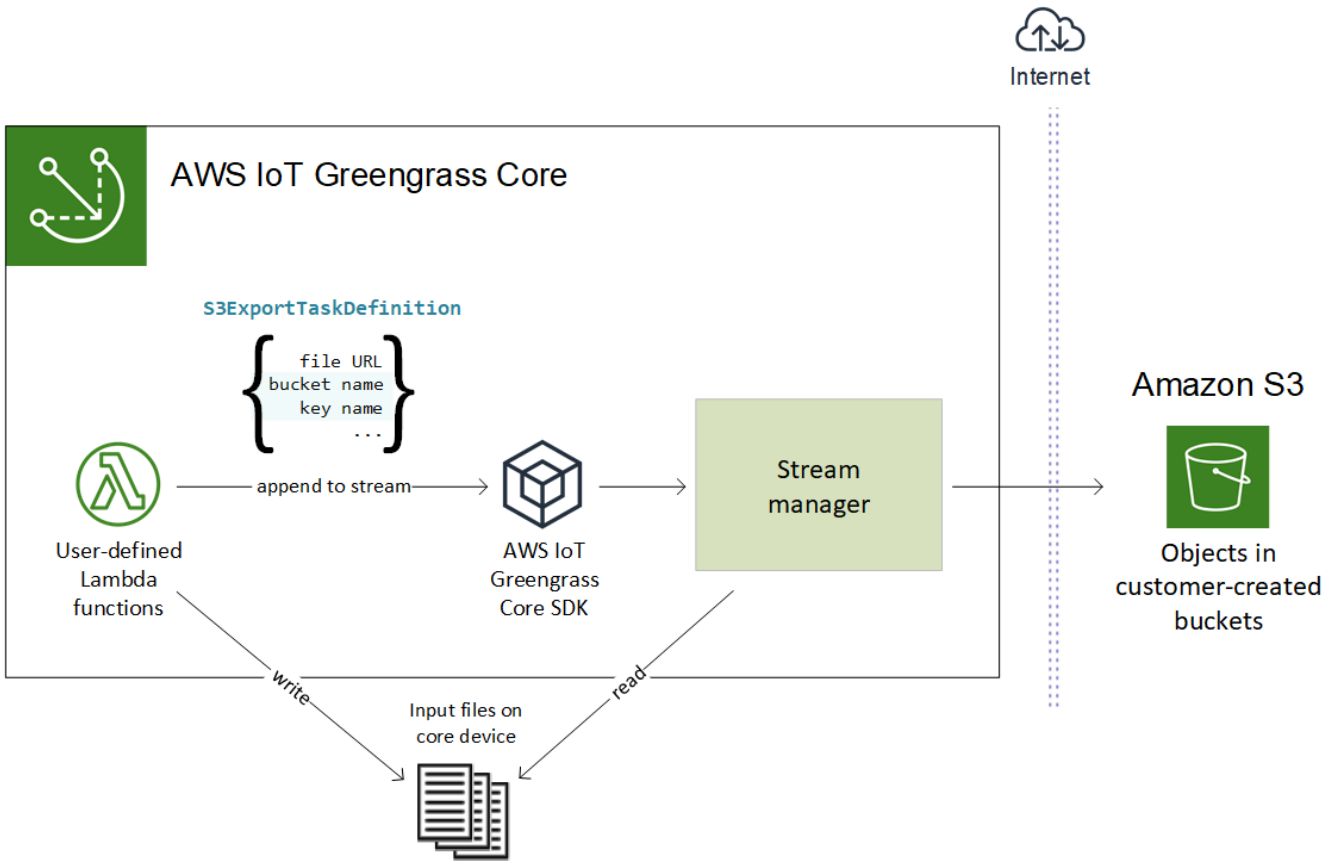
Anda dapat memberikan akses terperinci atau bersyarat ke sumber daya, misalnya dengan menggunakan skema penamaan wildcard *. Untuk informasi lebih lanjut, lihat Menambahkan dan menghapus kebijakan IAM dalam Panduan Pengguna IAM.

Mengekspor ke Amazon S3

Untuk membuat pengaliran yang mengekspor ke Amazon S3, fungsi Lambda Anda menggunakan `S3ExportTaskExecutorConfig` objek untuk mengonfigurasi kebijakan ekspor. Kebijakan ini mendefinisikan pengaturan ekspor, seperti ambang dan prioritas unggahan multipart. Untuk ekspor Amazon S3, pengelola pengaliran mengunggah data yang dibaca dari file lokal pada perangkat core. Untuk memulai unggahan, fungsi Lambda Anda menambahkan tugas ekspor ke pengaliran target. Tugas ekspor berisi informasi tentang file input dan objek Amazon S3 target. Pengelola pengaliran mengeksekusi tugas dalam urutan yang mereka tambahkan ke pengaliran.

> **Note**
>
> Target bucket harus sudah ada dalam Akun AWS. Jika sebuah objek untuk kunci tertentu tidak ada, pengelola pengaliran membuat objek untuk Anda.

Alur kerja tingkat tinggi ini ditunjukkan dalam diagram berikut.



Pengelola pengaliran menggunakan unggahan multipart ambang properti, ukuran bagian minimum pengaturan, dan ukuran file input untuk menentukan cara untuk mengunggah data. Ambang batas unggahan multipart harus lebih besar atau sama dengan ukuran bagian minimum. Jika Anda ingin meng-upload data secara paralel, Anda dapat membuat beberapa aliran.

Kunci yang menentukan objek Amazon S3 target Anda dapat mencakup string Java DateTimeFormatter yang valid di placeholder `!{timestamp:value}`. Anda dapat menggunakan placeholder stempel waktu ini untuk membagi data di Amazon S3 berdasarkan waktu data file input tersebut diunggah. Sebagai contoh, nama kunci berikut memutuskan untuk nilai seperti `my-key/2020/12/31/data.txt`.

```
my-key/!{timestamp:YYYY}/!{timestamp:MM}/!{timestamp:dd}/data.txt
```

> **Note**
>
> Jika Anda ingin memantau status ekspor untuk suatu aliran, pertama-tama buat status aliran dan kemudian konfigurasi aliran ekspor untuk menggunakannya. Untuk informasi selengkapnya, lihat the section called “Pantau tugas ekspor”.

Kelola data input

Anda dapat penulis kode yang digunakan aplikasi IoT untuk mengelola siklus hidup dari input data. Contoh alur kerja berikut menunjukkan cara Anda mungkin menggunakan fungsi Lambda untuk mengelola data ini.

1. Proses lokal menerima data dari perangkat atau periferal, lalu menulis data ke file dalam direktori pada perangkat core. Ini adalah file input untuk pengelola pengaliran.

   > **Note**
   >
   > Untuk menentukan apakah Anda harus mengonfigurasi akses ke direktori file input, lihat parameter STREAM_MANAGER_READ_ONLY_DIRS ini.
   > Proses yang menjalankan pengelola pengaliran dalam mewarisi semua izin sistem file identitas akses default untuk grup. Pengelola pengaliran harus memiliki izin untuk mengakses file input. Anda dapat menggunakan `chmod(1)` perintah untuk mengubah izin dari file, jika perlu.

2. Sebuah fungsi Lambda memindai direktori dan menambahkan tugas ekspor ke pengaliran target ketika file baru dibuat. Tugas adalah JSON-serialized `S3ExportTaskDefinition` objek yang menentukan URL dari file input, bucket dan kunci Amazon S3 target, dan metadata pengguna opsional.
3. Stream manager membaca file input dan mengekspor data ke Amazon S3 dalam urutan tugas yang ditambahkan. Bucket target harus sudah ada dalam perangkat Akun AWS Anda. Jika sebuah objek untuk kunci tertentu tidak ada, pengelola pengaliran membuat objek untuk Anda.
4. Fungsi Lambda membaca pesan dari pengaliran status untuk memantau status ekspor. Setelah tugas ekspor selesai, fungsi Lambda dapat menghapus file input yang sesuai. Untuk informasi selengkapnya, lihat the section called “Pantau tugas ekspor”.

Pantau tugas ekspor

Anda dapat penulis kode yang digunakan aplikasi IoT untuk memantau status ekspor Amazon S3 Anda. Fungsi Lambda Anda harus membuat status pengaliran lalu mengonfigurasi pengaliran ekspor untuk menulis pembaruan ke status untuk status pengaliran. Pengaliran status tunggal dapat menerima pembaruan status dari beberapa pengaliran yang mengekspor ke Amazon S3.

Pertama-tama, buat aliran yang akan digunakan sebagai status aliran. Anda dapat mengonfigurasi kebijakan ukuran dan penyimpanan bagi aliran tersebut untuk mengontrol umur pesan status. Misalnya:

- Tetapkan `Persistence` ke `Memory` jika Anda tidak ingin menyimpan pesan status.
- Tetapkan `StrategyOnFull` ke `OverwriteOldestData` agar pesan status baru tidak hilang.

Kemudian, buat atau perbarui aliran ekspor untuk menggunakan status aliran. Secara khusus, atur properti konfigurasi status dari konfigurasi ekspor `S3ExportTaskExecutorConfig` pengaliran. Ini memberitahu pengelola pengaliran untuk menulis pesan status tentang tugas ekspor ke pengaliran status. Di `StatusConfig` objek, tentukan nama pengaliran status dan tingkat verbositas. Nilai yang didukung berikut berkisar dari verbositas paling kecil (`ERROR`) hingga verbositas tertinggi (`TRACE`). Default-nya adalah `INFO`.

- `ERROR`
- `WARN`
- `INFO`
- `DEBUG`
- `TRACE`

Contoh alur kerja berikut menunjukkan cara fungsi Lambda mungkin menggunakan status pengaliran untuk memantau status ekspor.

1. Seperti yang dijelaskan dalam alur kerja sebelumnya, fungsi Lambda menambahkan tugas ekspor ke pengaliran yang dikonfigurasi untuk menulis pesan status tentang tugas ekspor ke pengaliran status. Operasi penambahan mengembalikan nomor urut yang mewakili ID tugas.
2. Sebuah fungsi Lambda membaca pesan secara berurutan dari pengaliran status, lalu menyaring pesan berdasarkan nama pengaliran dan ID tugas atau berdasarkan properti tugas ekspor dari

konteks pesan. Sebagai contoh, fungsi Lambda dapat difilter oleh URL file input dari tugas ekspor, yang diwakili oleh `S3ExportTaskDefinition` objek dalam konteks pesan.

Kode status berikut menunjukkan bahwa tugas ekspor telah mencapai keadaan selesai:

- `Success`. Upload berhasil diselesaikan.
- `Failure`. Stream manager mengalami kesalahan, misalnya, bucket yang ditentukan tidak ada. Setelah menyelesaikan masalah, Anda dapat menambahkan tugas ekspor ke aliran lagi.
- `Canceled`. Tugas dibatalkan karena definisi aliran atau ekspor dihapus, atau time-to-live (TTL) periode tugas berakhir.

> **Note**
>
> Tugas mungkin juga memiliki status `InProgress` atau `Warning`. Pengelola pengaliran mengeluarkan peringatan ketika suatu peristiwa mengembalikan kesalahan yang tidak memengaruhi pelaksanaan tugas. Sebagai contoh, kegagalan untuk membersihkan unggahan parsial dibatalkan mengembalikan peringatan.

3. Setelah tugas ekspor selesai, fungsi Lambda dapat menghapus file input yang sesuai.

Contoh berikut menunjukkan cara fungsi Lambda mungkin membaca dan memproses pesan status.

Python

```
import time
from greengrasssdk.stream_manager import (
    ReadMessagesOptions,
    Status,
    StatusConfig,
    StatusLevel,
    StatusMessage,
    StreamManagerClient,
)
from greengrasssdk.stream_manager.util import Util

client = StreamManagerClient()

try:
    # Read the statuses from the export status stream
    is_file_uploaded_to_s3 = False
    while not is_file_uploaded_to_s3:
```

```
        try:
            messages_list = client.read_messages(
                "StatusStreamName", ReadMessagesOptions(min_message_count=1,
 read_timeout_millis=1000)
            )
            for message in messages_list:
                # Deserialize the status message first.
                status_message = Util.deserialize_json_bytes_to_obj(message.payload,
 StatusMessage)

                # Check the status of the status message. If the status is
 "Success",
                # the file was successfully uploaded to S3.
                # If the status was either "Failure" or "Cancelled", the server was
 unable to upload the file to S3.
                # We will print the message for why the upload to S3 failed from the
 status message.
                # If the status was "InProgress", the status indicates that the
 server has started uploading
                # the S3 task.
                if status_message.status == Status.Success:
                    logger.info("Successfully uploaded file at path " + file_url + "
 to S3.")
                    is_file_uploaded_to_s3 = True
                elif status_message.status == Status.Failure or
 status_message.status == Status.Canceled:
                    logger.info(
                        "Unable to upload file at path " + file_url + " to S3.
 Message: " + status_message.message
                    )
                    is_file_uploaded_to_s3 = True
            time.sleep(5)
        except StreamManagerException:
            logger.exception("Exception while running")
except StreamManagerException:
    pass
    # Properly handle errors.
except ConnectionError or asyncio.TimeoutError:
    pass
    # Properly handle errors.
```

Referensi Python SDK: baca_pesan | StatusPesan

Java

```
import com.amazonaws.greengrass.streammanager.client.StreamManagerClient;
import com.amazonaws.greengrass.streammanager.client.utils.ValidateAndSerialize;
import com.amazonaws.greengrass.streammanager.model.ReadMessagesOptions;
import com.amazonaws.greengrass.streammanager.model.Status;
import com.amazonaws.greengrass.streammanager.model.StatusConfig;
import com.amazonaws.greengrass.streammanager.model.StatusLevel;
import com.amazonaws.greengrass.streammanager.model.StatusMessage;

try (final StreamManagerClient client =
 GreengrassClientBuilder.streamManagerClient().build()) {
    try {
        boolean isS3UploadComplete = false;
        while (!isS3UploadComplete) {
            try {
                // Read the statuses from the export status stream
                List<Message> messages = client.readMessages("StatusStreamName",
                    new
 ReadMessagesOptions().withMinMessageCount(1L).withReadTimeoutMillis(1000L));
                for (Message message : messages) {
                    // Deserialize the status message first.
                    StatusMessage statusMessage =
 ValidateAndSerialize.deserializeJsonBytesToObj(message.getPayload(),
 StatusMessage.class);
                    // Check the status of the status message. If the status is
 "Success", the file was successfully uploaded to S3.
                    // If the status was either "Failure" or "Canceled", the server
 was unable to upload the file to S3.
                    // We will print the message for why the upload to S3 failed
 from the status message.
                    // If the status was "InProgress", the status indicates that the
 server has started uploading the S3 task.
                    if (Status.Success.equals(statusMessage.getStatus())) {
                        System.out.println("Successfully uploaded file at path " +
 FILE_URL + " to S3.");
                        isS3UploadComplete = true;
                     } else if (Status.Failure.equals(statusMessage.getStatus()) ||
 Status.Canceled.equals(statusMessage.getStatus())) {
                        System.out.println(String.format("Unable to upload file at
 path %s to S3. Message %s",

 statusMessage.getStatusContext().getS3ExportTaskDefinition().getInputUrl(),
                            statusMessage.getMessage()));
```

```
                        sS3UploadComplete = true;
                    }
                }
            } catch (StreamManagerException ignored) {
            } finally {
                // Sleep for sometime for the S3 upload task to complete before
 trying to read the status message.
                Thread.sleep(5000);
            }
        } catch (e) {
        // Properly handle errors.
    }
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Referensi SDK Java: readMessages | StatusMessage

Node.js

```
const {
    StreamManagerClient, ReadMessagesOptions,
    Status, StatusConfig, StatusLevel, StatusMessage,
    util,
} = require('aws-greengrass-core-sdk').StreamManager;

const client = new StreamManagerClient();
client.onConnected(async () => {
    try {
        let isS3UploadComplete = false;
        while (!isS3UploadComplete) {
            try {
                // Read the statuses from the export status stream
                const messages = await c.readMessages("StatusStreamName",
                    new ReadMessagesOptions()
                        .withMinMessageCount(1)
                        .withReadTimeoutMillis(1000));

                messages.forEach((message) => {
                    // Deserialize the status message first.
                    const statusMessage =
 util.deserializeJsonBytesToObj(message.payload, StatusMessage);
                    // Check the status of the status message. If the status is
 'Success', the file was successfully uploaded to S3.
```

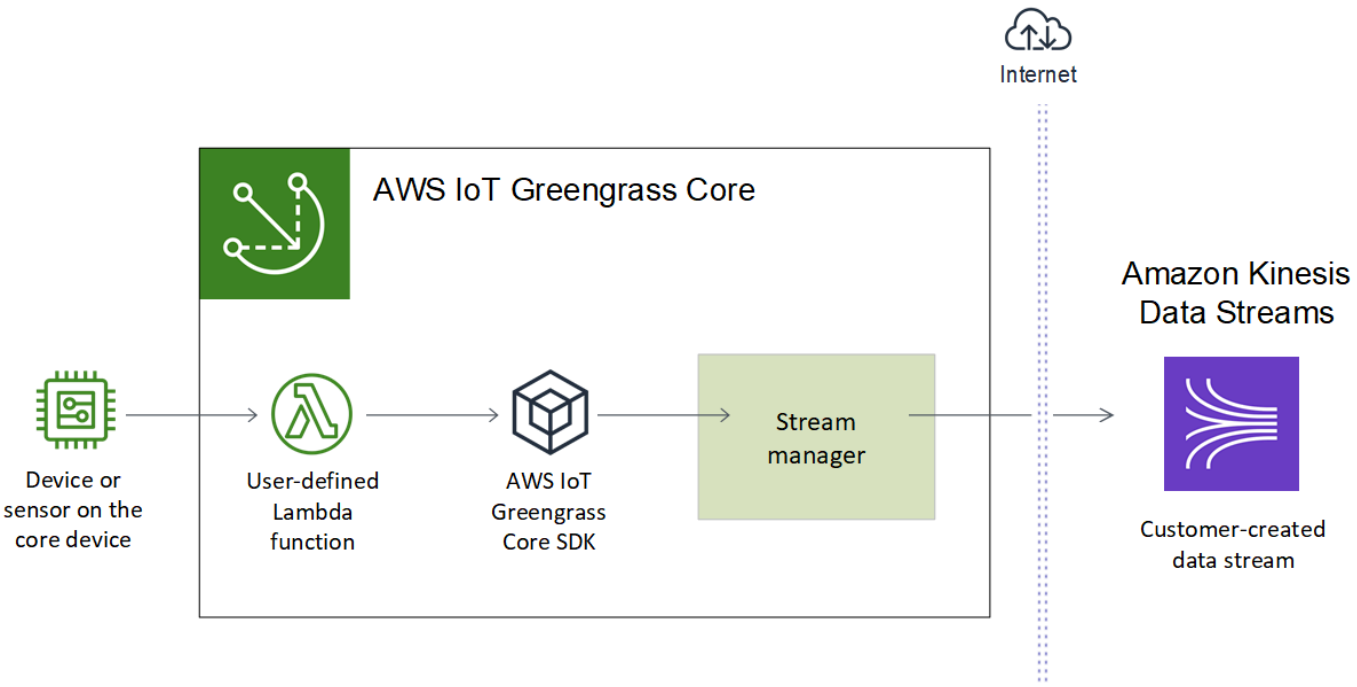
```
                    // If the status was either 'Failure' or 'Cancelled', the server
 was unable to upload the file to S3.
                    // We will print the message for why the upload to S3 failed
 from the status message.
                    // If the status was "InProgress", the status indicates that the
 server has started uploading the S3 task.
                    if (statusMessage.status === Status.Success) {
                        console.log(`Successfully uploaded file at path ${FILE_URL}
 to S3.`);
                        isS3UploadComplete = true;
                    } else if (statusMessage.status === Status.Failure ||
 statusMessage.status === Status.Canceled) {
                        console.log(`Unable to upload file at path ${FILE_URL} to
 S3. Message: ${statusMessage.message}`);
                        isS3UploadComplete = true;
                    }
                });
                // Sleep for sometime for the S3 upload task to complete before
 trying to read the status message.
                await new Promise((r) => setTimeout(r, 5000));
            } catch (e) {
                // Ignored
            }
    } catch (e) {
        // Properly handle errors.
    }
});
client.onError((err) => {
    // Properly handle connection errors.
    // This is called only when the connection to the StreamManager server fails.
});
```

Referensi SDK Node.js: readMessages | StatusMessage

Ekspor aliran data ke AWS Cloud (konsol)

Tutorial ini menunjukkan cara menggunakan AWS IoT konsol untuk mengonfigurasi dan men-deploy AWS IoT Greengrass dengan pengelola pengaliran yang diaktifkan. Grup ini berisi fungsi Lambda yang ditetapkan pengguna yang menulis ke pengaliran di pengelola pengaliran, yang kemudian diekspor secara otomatis ke AWS Cloud.

Pengelola pengaliran membuat penyerapan, pemrosesan, dan ekspor pengaliran data volume tinggi menjadi lebih efisien dan andal. Dalam tutorial ini, Anda membuat `TransferStream` fungsi Lambda yang mengonsumsi data IoT. Fungsi Lambda menggunakan AWS IoT Greengrass Core SDK untuk membuat pengaliran di pengelola pengaliran lalu membaca dan menulis untuk itu. Pengelola pengaliran kemudian mengekspor pengaliran ke Kinesis Data Streams. Diagram berikut menunjukkan alur kerja ini.



Fokus dari tutorial ini adalah untuk menunjukkan cara fungsi Lambda yang ditetapkan pengguna menggunakan objek `StreamManagerClient` dalam AWS IoT Greengrass Core SDK untuk berinteraksi dengan pengelola pengaliran. Untuk mempermudah, fungsi Lambda Python yang Anda buat untuk tutorial ini menghasilkan data perangkat simulasi.

Prasyarat

Untuk menyelesaikan tutorial ini, Anda memerlukan:

- Sebuah grup Greengrass dan core Greengrass (v1.10 atau yang lebih baru). Untuk informasi tentang cara membuat grup Greengrass dan core, lihat Memulai dengan AWS IoT Greengrass. Tutorial Memulai juga mencakup langkah-langkah untuk menginstal AWS IoT Greengrass perangkat lunak Core.

> **Note**
>
> Pengelola pengaliran tidak didukung OpenWrt distribusi.

- Waktu aktif Java 8 (JDK 8) dipasang pada perangkat core.
  - Untuk distribusi berbasis Debian (termasuk Raspbian) atau distribusi berbasis Ubuntu, jalankan perintah berikut:

    ```
    sudo apt install openjdk-8-jdk
    ```

  - Untuk distribusi berbasis Red Hat (termasuk Amazon Linux), jalankan perintah berikut:

    ```
    sudo yum install java-1.8.0-openjdk
    ```

    Untuk informasi lebih lanjut, lihat Cara mengunduh dan menginstal paket OpenJDK prebuilt dalam dokumentasi OpenJDK.
- AWS IoT Greengrass Core SDK for Python v1.5.0 atau yang lebih baru. Untuk menggunakan `StreamManagerClient` dalam AWS IoT Greengrass Core SDK for Python, Anda harus:
  - Instal Python 3.7 atau yang lebih baru pada perangkat core.
  - Sertakan SDK dan dependensinya dalam paket deployment fungsi Lambda Anda. Instruksi disediakan pada tutorial ini.

> **Tip**
>
> Anda dapat menggunakan `StreamManagerClient` dengan Java atau NodeJS. Untuk kode sampel sampel, lihatAWS IoT GreengrassCore SDK for JavadanAWS IoT GreengrassCore SDK for Node.jsdi atas GitHub.

- Pengaliran tujuan bernama **`MyKinesisStream`** dibuat di Amazon Kinesis Data Streams dalam hal yang sama Wilayah AWS sebagai grup Greengrass Anda. Untuk informasi lebih lanjut, lihat Buat aliran dalam Panduan Developer Amazon Kinesis.

> **Note**
>
> Dalam tutorial ini, pengelola pengaliran mengekspor data ke Kinesis Data Streams, yang mengakibatkan biaya ke Akun AWS. Untuk informasi lebih lanjut tentang harga, lihat Harga Kinesis Data Streams.
>
> Untuk menghindari timbulnya biaya, Anda dapat menjalankan tutorial ini tanpa membuat Kinesis data stream. Dalam kasus ini, Anda memeriksa catatan untuk melihat pengelola pengaliran tersebut berusaha mengekspor pengaliran ke Kinesis Data Streams.

- Kebijakan IAM ditambahkan ke the section called “Peran grup Greengrass” yang mengizinkan tindakan `kinesis:PutRecords` pada aliran data target, seperti yang ditunjukkan dalam contoh berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:PutRecords"
            ],
            "Resource": [
                "arn:aws:kinesis:region:account-id:stream/MyKinesisStream"
            ]
        }
    ]
}
```

Tutorial berisi langkah-langkah tingkat tinggi berikut:

1. Buat paket deployment fungsi Lambda
2. Buat fungsi Lambda
3. Tambahkan fungsi ke grup
4. Mengaktifkan pengelola pengaliran
5. Konfigurasikan logging lokal
6. Men-deploy grup
7. Uji aplikasi

Tutorial akan memakan waktu sekitar 20 menit untuk menyelesaikannya.

Langkah 1: Buat paket deployment fungsi Lambda

Dalam langkah ini, Anda membuat paket deployment fungsi Lambda yang berisi kode fungsi Python dan dependensi. Anda mengunggah paket ini nanti ketika Anda membuat fungsi Lambda dalam AWS Lambda. Fungsi Lambda menggunakan AWS IoT Greengrass Core SDK untuk membuat dan berinteraksi dengan pengaliran lokal.

> **Note**
>
> Fungsi Lambda yang ditetapkan pengguna milik Anda harus menggunakan AWS IoT Greengrass Core SDK untuk berinteraksi dengan pengelola pengaliran. Untuk informasi lebih lanjut tentang persyaratan untuk Greengrass pengelola pengaliran, lihat Persyaratan pengelola pengaliran Greengrass.

1. Unduh AWS IoT Greengrass Core SDK for Python v1.5.0 atau yang lebih baru.
2. Unzip paket unduhan untuk mendapatkan SDK. SDK adalah folder `greengrasssdk` ini.
3. Instal paket dependensi untuk menyertakan dengan SDK dalam paket deployment fungsi Lambda Anda.

   1. Arahkan ke direktori SDK yang berisi file `requirements.txt` ini. File ini berisi daftar dependensi.
   2. Instal dependensi SDK. Sebagai contoh, jalankan perintah `pip` berikut untuk menginstalnya di direktori saat ini:

      ```
      pip install --target . -r requirements.txt
      ```
4. Simpan fungsi kode Python berikut dalam sebuah file lokal bernama `transfer_stream.py`.

   > **Tip**
   >
   > Misalnya kode yang menggunakan Java dan NodeJS, lihatAWS IoT GreengrassCore SDK for JavadanAWS IoT GreengrassCore SDK for Node.jsdi atas GitHub.

   ```
   import asyncio
   ```

```
import logging
import random
import time

from greengrasssdk.stream_manager import (
    ExportDefinition,
    KinesisConfig,
    MessageStreamDefinition,
    ReadMessagesOptions,
    ResourceNotFoundException,
    StrategyOnFull,
    StreamManagerClient,
)


# This example creates a local stream named "SomeStream".
# It starts writing data into that stream and then stream manager automatically
 exports
# the data to a customer-created Kinesis data stream named "MyKinesisStream".
# This example runs forever until the program is stopped.

# The size of the local stream on disk will not exceed the default (which is 256
 MB).
# Any data appended after the stream reaches the size limit continues to be
 appended, and
# stream manager deletes the oldest data until the total stream size is back under
 256 MB.
# The Kinesis data stream in the cloud has no such bound, so all the data from this
 script is
# uploaded to Kinesis and you will be charged for that usage.


def main(logger):
    try:
        stream_name = "SomeStream"
        kinesis_stream_name = "MyKinesisStream"

        # Create a client for the StreamManager
        client = StreamManagerClient()

        # Try deleting the stream (if it exists) so that we have a fresh start
        try:
            client.delete_message_stream(stream_name=stream_name)
        except ResourceNotFoundException:
```

```
            pass

        exports = ExportDefinition(
            kinesis=[KinesisConfig(identifier="KinesisExport" + stream_name,
kinesis_stream_name=kinesis_stream_name)]
        )
        client.create_message_stream(
            MessageStreamDefinition(
                name=stream_name,
strategy_on_full=StrategyOnFull.OverwriteOldestData, export_definition=exports
            )
        )

        # Append two messages and print their sequence numbers
        logger.info(
            "Successfully appended message to stream with sequence number %d",
            client.append_message(stream_name, "ABCDEFGHIJKLMNO".encode("utf-8")),
        )
        logger.info(
            "Successfully appended message to stream with sequence number %d",
            client.append_message(stream_name, "PQRSTUVWXYZ".encode("utf-8")),
        )

        # Try reading the two messages we just appended and print them out
        logger.info(
            "Successfully read 2 messages: %s",
            client.read_messages(stream_name,
ReadMessagesOptions(min_message_count=2, read_timeout_millis=1000)),
        )

        logger.info("Now going to start writing random integers between 0 and 1000
to the stream")
        # Now start putting in random data between 0 and 1000 to emulate device
sensor input
        while True:
            logger.debug("Appending new random integer to stream")
            client.append_message(stream_name, random.randint(0,
1000).to_bytes(length=4, signed=True, byteorder="big"))
            time.sleep(1)

    except asyncio.TimeoutError:
        logger.exception("Timed out while executing")
    except Exception:
        logger.exception("Exception while running")
```

```
def function_handler(event, context):
    return


logging.basicConfig(level=logging.INFO)
# Start up this sample code
main(logger=logging.getLogger())
```

5. Zip item berikut ke dalam file bernama `transfer_stream_python.zip`. Ini adalah paket deployment fungsi Lambda Anda.

   - transfer_stream.py. Logika aplikasi.
   - greengrasssdk. Diperlukan perpustakaan fungsi Python Greengrass Lambda yang menerbitkankan pesan MQTT.

     Operasi pengelola pengaliran tersedia dalam versi 1.5.0 atau yang lebih baru dari AWS IoT Greengrass Core SDK for Python.
   - Dependensi yang Anda instal untuk AWS IoT Greengrass Core SDK for Python (sebagai contoh, direktori `cbor2` ini).

   Ketika Anda membuat file `zip` ini, termasuk hanya item ini, bukan folder yang berisi.

Langkah 2: Buat fungsi Lambda

Pada langkah ini, Anda menggunakan konsol AWS Lambda untuk membuat fungsi Lambda dan mengonfigurasinya agar menggunakan paket deployment Anda. Kemudian, Anda mempublikasikan versi fungsi dan membuat alias.

1. Pertama, buat fungsi Lambda.

   a. Di AWS Management Console, pilih Layanan, dan buka konsol AWS Lambda tersebut.

   b. Pilih Buat fungsi dan kemudian Tulis dari awal.

   c. Di bagian Informasi dasar tersebut, gunakan nilai-nilai berikut:

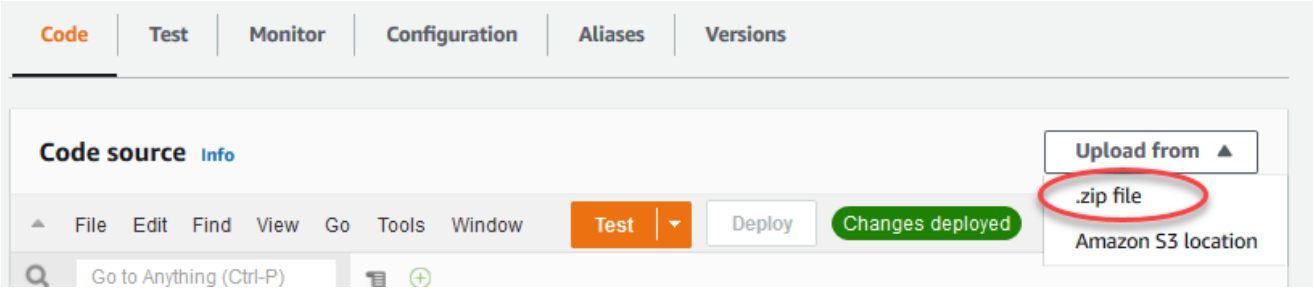
      - Untuk Nama fungsi, masukkan **`TransferStream`**.
      - Untuk Waktu pengoperasian, pilih Python 3.7.

- Untuk Izin, pertahankan pengaturan default. Hal ini menciptakan peran eksekusi yang memberikan izin Lambda basic. Peran ini tidak digunakan oleh AWS IoT Greengrass.

d. Di bagian bawah halaman, pilih Buat Fungsi.

2. Selanjutnya, daftarkan handler dan unggah paket deployment fungsi Lambda Anda.

a. Pada tab Kode ini, di bawah Sumber kode, pilih Unggah dari. Dari dropdown, pilih file .zip.



b. Pilih Unggah, lalu pilih paket deployment `transfer_stream_python.zip` Anda. Lalu, pilih Simpan.

c. Pada tab Kode fungsi, di bawah Pengaturan waktu aktif, pilih Edit, dan kemudian masukkan nilai-nilai berikut.

- Untuk Waktu pengoperasian, pilih Python 3.7.
- Untuk Handler, masukkan **`transfer_stream.function_handler`**

d. Pilih Save (Simpan).

> **Note**
>
> Tombol Tes pada konsol AWS Lambda tidak bekerja dengan fungsi ini. Pada AWS IoT Greengrass Core SDK tidak berisi modul yang diperlukan untuk menjalankan fungsi Greengrass Lambda Anda secara independen di konsol AWS Lambda tersebut. Modul-modul ini (misalnya, `greengrass_common`) dipasok ke fungsi setelah mereka di-deploy ke core Greengrass Anda.

3. Sekarang, publikasikan versi pertama fungsi Lambda Anda dan membuat alias untuk versi.

> **Note**
>
> Grup Greengrass dapat mereferensi fungsi Lambda dengan alias (direkomendasikan) atau dengan versi. Menggunakan alias membuatnya lebih mudah untuk mengelola pembaruan kode karena Anda tidak perlu mengubah tabel langganan atau definisi grup

ketika kode fungsi diperbarui. Sebaliknya, Anda hanya mengarahkan alias ke versi fungsi baru.

a. Dari menu Tindakan ini, pilih Terbitkan versi baru.
b. Untuk Versi Deskripsi, masukkan **`First version`**, lalu pilih Publikasikan.
c. PadaTransferStream: 1halaman konfigurasi, dariTindakanmenu, pilihMembuat alias.
d. Pada halaman Buat alias baru ini, gunakan nilai-nilai berikut:

   - Untuk Nama, masukkan **`GG_TransferStream`**.
   - Untuk Versi, pilih 1.

   > **Note**
   >
   > AWS IoT Greengrass tidak support alias Lambda untuk versi $TERBARU ini.

e. Pilih Create (Buat).

Sekarang Anda siap untuk menambahkan fungsi Lambda ke grup Greengrass Anda.

Langkah 3: Tambahkan fungsi Lambda ke grup Greengrass

Dalam langkah ini, Anda menambahkan fungsi Lambda ke grup lalu mengonfigurasi siklus hidup dan lingkungan variabel. Untuk informasi selengkapnya, lihat the section called “Mengontrol eksekusi fungsi Greengrass Lambda”.

1. DiAWS IoTpanel navigasi konsol, di bawahKelola, PerluasPerangkat Greengrass, lalu pilihGrup (V1).
2. Pilih grup target.
3. Pada halaman konfigurasi grup, pilihFungsi Lambdatab.
4. Di bawahFungsi Lambda saya, pilihTambahkan.
5. PadaTambahkan fungsi Lambdahalaman, pilihFungsi Lambdauntuk fungsi Lambda Anda.
6. UntukVersi Lambda, pilihAlias: GG_TransferStream.

   Sekarang, konfigurasikan properti yang menentukan perilaku fungsi Lambda dalam grup Greengrass.

7. DiKonfigurasi fungsi Lambdabagian, lakukan perubahan berikut:

   - Atur Batas memori hingga 32 MB.
   - UntukDi-pin, pilihBenar.

   > **Note**
   >
   > Fungsi Lambda berumur panjang (atau disematkan) memulai secara otomatis setelah AWS IoT Greengrass dimulai dan terus berjalan dalam kontainernya sendiri. Hal ini berbeda dengan fungsi Lambda sesuai permintaan ini, yang dimulai ketika diaktifkan dan berhenti ketika tidak ada tugas yang tersisa untuk dijalankan. Untuk informasi selengkapnya, lihat the section called “Konfigurasi siklus hidup”.

8. PilihTambahkan fungsi Lambda.

Langkah 4: Mengaktifkan pengelola pengaliran

Dalam langkah ini, Anda memastikan bahwa pengelolal pengaliran diaktifkan.

1. Pada halaman konfigurasi grup, pilihFungsi Lambdatab.
2. Di bawahFungsi Lambda sistem, pilihManajer pengaliran, dan periksa statusnya. Jika dinonaktifkan, pilih Edit. Kemudian, pilih Aktifkan dan Simpan. Anda dapat menggunakan pengaturan parameter default untuk tutorial ini. Untuk informasi selengkapnya, lihat the section called “Konfigurasi pengelola pengaliran”.

> **Note**
>
> Ketika Anda menggunakan konsol untuk mengaktifkan pengelola pengaliran dan men-deploy grup, ukuran memori untuk pengelola pengaliran diatur ke 4194304 KB (4 GB) secara default. Kami merekomendasikan Anda mengatur ukuran memori ke setidaknya 128000 KB.

Langkah 5: Konfigurasikan logging lokal

Dalam langkah ini, Anda mengonfigurasi AWS IoT Greengrass komponen sistem, fungsi Lambda yang ditetapkan pengguna, dan konektor dalam grup untuk menulis catatan ke sistem file perangkat

core. Anda dapat menggunakan catatan untuk memecahkan masalah yang mungkin Anda alami. Untuk informasi selengkapnya, lihat the section called “Pemantauan dengan AWS IoT Greengrass log”.

1. Di bawah Konfigurasikan catatan lokal, periksa apakah pengelogan lokal dikonfigurasi.
2. Jika catatan tidak dikonfigurasi untuk komponen sistem Greengrass atau fungsi Lambda yang ditetapkan pengguna, pilih Edit.
3. PilihTingkat log fungsi Lambda penggunadanTingkat log sistem Greengrass.
4. Menjaga nilai default untuk tingkat pencatatan dan batas ruang disk, lalu pilih Simpan.

Langkah 6: Men-deploy grup Greengrass

Men-deploy grup ke perangkat core.

1. Pastikan bahwa core AWS IoT Greengrass sedang berjalan. Jalankan perintah berikut di terminal Raspberry Pi Anda, sesuai kebutuhan.

   a. Untuk memeriksa apakah daemon sedang berjalan:

   ```
   ps aux | grep -E 'greengrass.*daemon'
   ```

   Jika outputnya berisi entri `root` untuk `/greengrass/ggc/packages/`*`ggc-version`*`/bin/daemon`, maka daemon sedang berjalan.

   > **Note**
   >
   > Versi di jalur tergantung pada versi perangkat lunak AWS IoT Greengrass core yang diinstal pada perangkat core Anda.

   b. Untuk memulai daemon:

   ```
   cd /greengrass/ggc/core/
   sudo ./greengrassd start
   ```

2. Pada halaman konfigurasi grup, pilihDeploy.
3. a. DiFungsi Lambdatab, di bawahFungsi Lambda sistembagian, pilihDetektor IPdan pilihlahedit.

b. DiMengedit pengaturan detektor IPkotak dialog, pilihSecara otomatis mendeteksi dan mengganti titik akhir broker MQTT.

c. Pilih Save (Simpan).

Hal ini mengaktifkan perangkat untuk secara otomatis memperoleh informasi konektivitas untuk core, seperti alamat IP, DNS, dan nomor port. Deteksi otomatis direkomendasikan, namun AWS IoT Greengrass juga support titik akhir yang ditentukan secara manual. Anda hanya diminta untuk metode penemuan pertama kalinya bahwa grup di-deploy.

> **Note**
>
> Jika diminta, berikan izin untuk membuat Peran layanan Greengrass dan kaitkan dengan Akun AWS Anda pada Wilayah AWS. Peran ini memungkinkan AWS IoT Greengrass untuk mengakses sumber daya Anda di layanan AWS ini.

Halaman Deployment menampilkan timestamp deployment, ID versi, dan status. Setelah selesai, status yang ditampilkan untuk deployment seharusnyaCompleted (Lengkap).

Untuk bantuan penyelesaian masalah, lihat Memecahkan masalah.

Langkah 7: Uji aplikasi

Fungsi Lambda `TransferStream` menghasilkan data perangkat yang disimulasikan. Ini menulis data ke pengaliran yang diekspor oleh pengelola pengaliran ke Kinesis data stream target.

1. Di konsol Amazon Kinesis, di bawahKinesis data streams, pilihMyKinesisStream.

> **Note**
>
> Jika Anda menjalankan tutorial tanpa Kinesis data streams target, periksa berkas log untuk pengelola pengaliran (`GGStreamManager`). Jika itu berisi `export stream MyKinesisStream doesn't exist` dalam pesan kesalahan, maka uji ini berhasil. Kesalahan ini berarti bahwa layanan mencoba untuk mengekspor ke pengaliran tetapi pengaliran tidak ada.

2. PadaMyKinesisStreamhalaman, pilihPemantauan. Jika uji berhasil, Anda akan melihat data di bagan Pasang Catatan ini. Tergantung pada koneksi Anda, mungkin diperlukan waktu satu menit sebelum data ditampilkan.

   > **Important**
   >
   > Setelah selesai melakukan pengujian, hapus Kinesis data stream agar tidak menimbulkan biaya tambahan.
   > Atau, jalankan perintah berikut untuk menghentikan daemon Greengrass. Hal ini mencegah core mengirim pesan hingga Anda siap untuk melanjutkan pengujian.
   >
   > ```
   > cd /greengrass/ggc/core/
   > sudo ./greengrassd stop
   > ```

3. HapusTransferStreamFungsi Lambda dari inti.

   a. DiAWS IoTpanel navigasi konsol, di bawahKelola, PerluasPerangkat Greengrass, lalu pilihGrup (V1).

   b. Di bawah grup Greengrass, pilih grup Anda.

   c. PadaLambdahalaman, pilih elips (...) untukTransferStreamfungsi, dan kemudian pilihHapus fungsi.

   d. Dari Tindakan, pilih Men-deploy.

Untuk melihat pencatatan informasi atau pemecahan masalah dengan aliran, periksa log untuk fungsi `TransferStream` dan `GGStreamManager` tersebut. Anda harus memiliki izin `root` untuk membaca catatan AWS IoT Greengrass pada sistem file.

- `TransferStream` menulis entri log ke *`greengrass-root`*`/ggc/var/log/user/`*`region`*`/`*`account-id`*`/TransferStream.log`.
- `GGStreamManager` menulis entri log ke *`greengrass-root`*`/ggc/var/log/system/GGStreamManager.log`.

Jika Anda membutuhkan informasi pemecahan masalah lainnya, Anda dapat mengatur tingkat logging untuk log Lambda pengguna ke log Debug lalu men-deploy grup lagi.

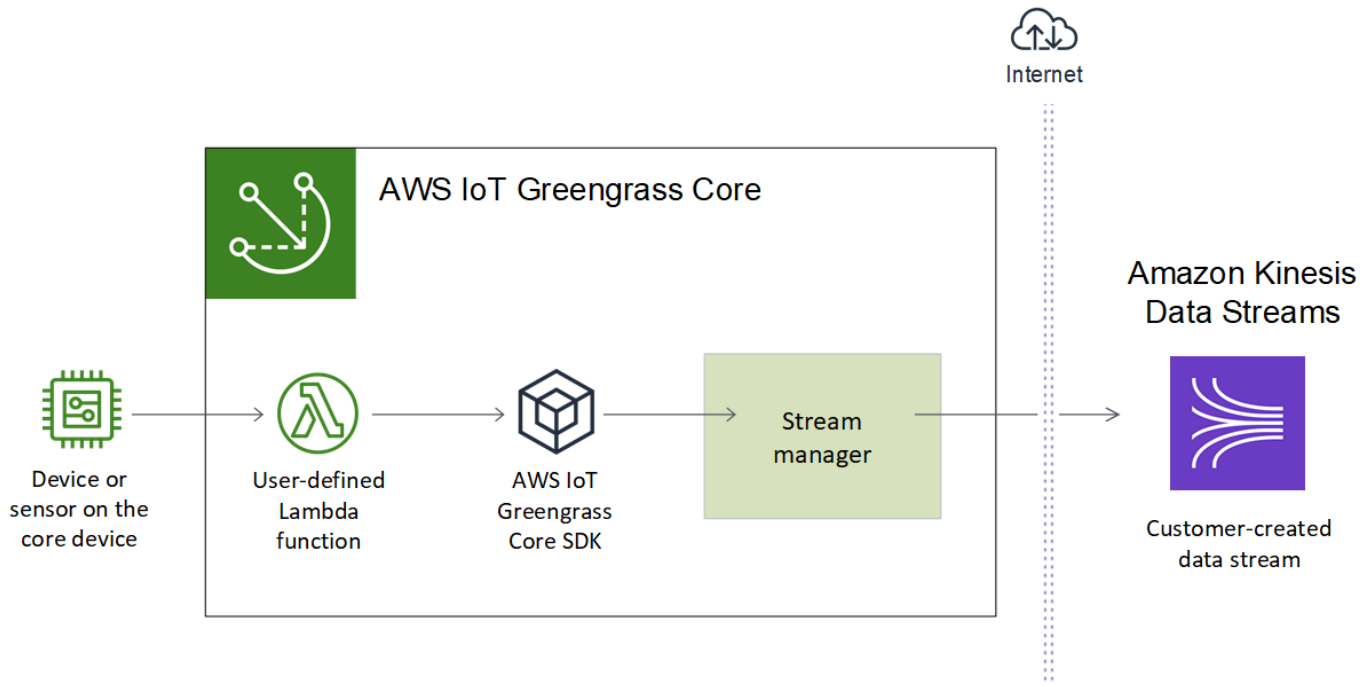
Lihat juga

- Mengelola aliran data
- the section called “Konfigurasi pengelola pengaliran”
- the section called “Gunakan StreamManagerClient untuk bekerja dengan aliran”
- the section called “Konfigurasi ekspor untuk tujuan AWS Cloud yang didukung”
- the section called “Ekspor aliran data (CLI)”

Ekspor aliran data ke AWS Cloud (CLI)

Tutorial ini menunjukkan bagaiman menggunakan AWS CLI untuk mengonfigurasi dan men-deploy AWS IoT Greengrass dengan pengelola pengaliran yang diaktifkan. Grup ini berisi fungsi Lambda yang ditetapkan pengguna yang menulis ke pengaliran di pengelola pengaliran, yang kemudian diekspor secara otomatis ke AWS Cloud.

Pengelola pengaliran membuat penyerapan, pemrosesan, dan ekspor pengaliran data volume tinggi menjadi lebih efisien dan andal. Dalam tutorial ini, Anda membuat `TransferStream` fungsi Lambda yang mengonsumsi data IoT. Fungsi Lambda menggunakan AWS IoT Greengrass Core SDK untuk membuat pengaliran di pengelola pengaliran lalu membaca dan menulis untuk itu. Pengelola pengaliran kemudian mengekspor pengaliran ke Kinesis Data Streams. Diagram berikut menunjukkan alur kerja ini.



Fokus dari tutorial ini adalah untuk menunjukkan cara fungsi Lambda yang ditetapkan pengguna menggunakan objek `StreamManagerClient` dalam AWS IoT Greengrass Core SDK untuk berinteraksi dengan pengelola pengaliran. Untuk mempermudah, fungsi Lambda Python yang Anda buat untuk tutorial ini menghasilkan data perangkat simulasi.

Saat Anda menggunakan AWS IoT Greengrass API, yang mencakup perintah Greengrass di AWS CLI, untuk membuat grup, pengelola aliran dinonaktifkan secara default. Untuk mengaktifkan pengelola pengaliran di core, Anda membuat versi definisi fungsi yang mencakup sistem `GGStreamManager` fungsi Lambda dan versi grup yang referensi versi definisi fungsi baru. Kemudian Anda men-deploy grup.

Prasyarat

Untuk menyelesaikan tutorial ini, Anda memerlukan:

- Sebuah grup Greengrass dan core Greengrass (v1.10 atau yang lebih baru). Untuk informasi tentang cara membuat grup Greengrass dan core, lihat Memulai dengan AWS IoT Greengrass. Tutorial Memulai juga mencakup langkah-langkah untuk menginstal AWS IoT Greengrass perangkat lunak Core.

  > **Note**
  >
  > Pengelola pengelola pengelola tidak didukung OpenWrt distribusi.

- Waktu aktif Java 8 (JDK 8) dipasang pada perangkat core.
  - Untuk distribusi berbasis Debian (termasuk Raspbian) atau distribusi berbasis Ubuntu, jalankan perintah berikut:

    ```
    sudo apt install openjdk-8-jdk
    ```

  - Untuk distribusi berbasis Red Hat (termasuk Amazon Linux), jalankan perintah berikut:

    ```
    sudo yum install java-1.8.0-openjdk
    ```

    Untuk informasi lebih lanjut, lihat Cara mengunduh dan menginstal paket OpenJDK prebuilt dalam dokumentasi OpenJDK.

- AWS IoT Greengrass Core SDK for Python v1.5.0 atau yang lebih baru. Untuk menggunakan `StreamManagerClient` dalam AWS IoT Greengrass Core SDK for Python, Anda harus:

- Instal Python 3.7 atau yang lebih baru pada perangkat core.
- Sertakan SDK dan dependensinya dalam paket deployment fungsi Lambda Anda. Instruksi disediakan pada tutorial ini.

> **Tip**
>
> Anda dapat menggunakan `StreamManagerClient` dengan Java atau NodeJS. Untuk kode sampel, lihatAWS IoT GreengrassCore SDK for JavadanAWS IoT GreengrassCore SDK for Node.jsdi atas GitHub.

- Pengaliran tujuan bernama **`MyKinesisStream`** dibuat di Amazon Kinesis Data Streams dalam hal yang sama Wilayah AWS sebagai grup Greengrass Anda. Untuk informasi lebih lanjut, lihat Buat aliran dalam Panduan Developer Amazon Kinesis.

> **Note**
>
> Dalam tutorial ini, pengelola pengaliran mengekspor data ke Kinesis Data Streams, yang mengakibatkan biaya ke Akun AWS. Untuk informasi lebih lanjut tentang harga, lihat Harga Kinesis Data Streams.
> Untuk menghindari timbulnya biaya, Anda dapat menjalankan tutorial ini tanpa membuat Kinesis data stream. Dalam kasus ini, Anda memeriksa catatan untuk melihat pengelola pengaliran tersebut berusaha mengekspor pengaliran ke Kinesis Data Streams.

- Kebijakan IAM ditambahkan ke the section called “Peran grup Greengrass” yang mengizinkan tindakan `kinesis:PutRecords` pada aliran data target, seperti yang ditunjukkan dalam contoh berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:PutRecords"
            ],
            "Resource": [
                "arn:aws:kinesis:region:account-id:stream/MyKinesisStream"
            ]
        }
```

```
    ]
}
```

- Menginstal AWS CLI dan mengonfigurasi ke komputer Anda. Untuk informasi lebih lanjut, lihat Menginstal AWS Command Line Interface dan Mengonfigurasi AWS CLI di AWS Command Line Interface Panduan Pengguna.

  Contoh perintah dalam tutorial ini ditulis untuk Linux dan sistem berbasis UNIX lainnya. Jika Anda menggunakan Windows, lihat Menentukan nilai parameter untuk AWS antarmuka baris perintah untuk informasi lebih lanjut tentang perbedaan dalam sintaks.

  Jika perintah berisi string JSON, tutorial memberikan contoh yang memiliki JSON pada satu baris. Pada beberapa sistem, mungkin lebih efisien untuk mengedit dan menjalankan perintah menggunakan format ini.

Tutorial berisi langkah-langkah tingkat tinggi berikut:

1. Buat paket deployment fungsi Lambda
2. Buat fungsi Lambda
3. Buat definisi fungsi dan versi
4. Buat definisi pencatat dan versi
5. Dapatkan ARN versi definisi core Anda
6. Buat versi grup
7. Buat deployment
8. Uji aplikasi

Tutorial harus memerlukan waktu sekitar 30 menit untuk menyelesaikannya.

Langkah 1: Buat paket deployment fungsi Lambda

Dalam langkah ini, Anda membuat paket deployment fungsi Lambda yang berisi kode fungsi Python dan dependensi. Anda mengunggah paket ini nanti ketika Anda membuat fungsi Lambda dalam

AWS Lambda. Fungsi Lambda menggunakan AWS IoT Greengrass Core SDK untuk membuat dan berinteraksi dengan pengaliran lokal.

> **Note**
>
> Fungsi Lambda yang ditetapkan pengguna milik Anda harus menggunakan AWS IoT Greengrass Core SDK untuk berinteraksi dengan pengelola pengaliran. Untuk informasi lebih lanjut tentang persyaratan untuk Greengrass pengelola pengaliran, lihat Persyaratan pengelola pengaliran Greengrass.

1. Unduh AWS IoT Greengrass Core SDK for Python v1.5.0 atau yang lebih baru.
2. Unzip paket unduhan untuk mendapatkan SDK. SDK adalah folder `greengrasssdk` ini.
3. Instal paket dependensi untuk menyertakan dengan SDK dalam paket deployment fungsi Lambda Anda.

   1. Arahkan ke direktori SDK yang berisi file `requirements.txt` ini. File ini berisi daftar dependensi.
   2. Instal dependensi SDK. Sebagai contoh, jalankan perintah `pip` berikut untuk menginstalnya di direktori saat ini:

      ```
      pip install --target . -r requirements.txt
      ```
4. Simpan fungsi kode Python berikut dalam sebuah file lokal bernama `transfer_stream.py`.

   > **Tip**
   >
   > Misalnya kode yang menggunakan Java dan NodeJS, lihatAWS IoT GreengrassCore SDK for JavadanAWS IoT GreengrassCore SDK for Node.jsdi atas GitHub.

   ```
   import asyncio
   import logging
   import random
   import time

   from greengrasssdk.stream_manager import (
       ExportDefinition,
   ```

```
    KinesisConfig,
    MessageStreamDefinition,
    ReadMessagesOptions,
    ResourceNotFoundException,
    StrategyOnFull,
    StreamManagerClient,
)


# This example creates a local stream named "SomeStream".
# It starts writing data into that stream and then stream manager automatically
 exports
# the data to a customer-created Kinesis data stream named "MyKinesisStream".
# This example runs forever until the program is stopped.

# The size of the local stream on disk will not exceed the default (which is 256
 MB).
# Any data appended after the stream reaches the size limit continues to be
 appended, and
# stream manager deletes the oldest data until the total stream size is back under
 256 MB.
# The Kinesis data stream in the cloud has no such bound, so all the data from this
 script is
# uploaded to Kinesis and you will be charged for that usage.


def main(logger):
    try:
        stream_name = "SomeStream"
        kinesis_stream_name = "MyKinesisStream"

        # Create a client for the StreamManager
        client = StreamManagerClient()

        # Try deleting the stream (if it exists) so that we have a fresh start
        try:
            client.delete_message_stream(stream_name=stream_name)
        except ResourceNotFoundException:
            pass

        exports = ExportDefinition(
            kinesis=[KinesisConfig(identifier="KinesisExport" + stream_name,
 kinesis_stream_name=kinesis_stream_name)]
        )
```

```
        client.create_message_stream(
            MessageStreamDefinition(
                name=stream_name,
 strategy_on_full=StrategyOnFull.OverwriteOldestData, export_definition=exports
            )
        )

        # Append two messages and print their sequence numbers
        logger.info(
            "Successfully appended message to stream with sequence number %d",
            client.append_message(stream_name, "ABCDEFGHIJKLMNO".encode("utf-8")),
        )
        logger.info(
            "Successfully appended message to stream with sequence number %d",
            client.append_message(stream_name, "PQRSTUVWXYZ".encode("utf-8")),
        )

        # Try reading the two messages we just appended and print them out
        logger.info(
            "Successfully read 2 messages: %s",
            client.read_messages(stream_name,
 ReadMessagesOptions(min_message_count=2, read_timeout_millis=1000)),
        )

        logger.info("Now going to start writing random integers between 0 and 1000
 to the stream")
        # Now start putting in random data between 0 and 1000 to emulate device
 sensor input
        while True:
            logger.debug("Appending new random integer to stream")
            client.append_message(stream_name, random.randint(0,
 1000).to_bytes(length=4, signed=True, byteorder="big"))
            time.sleep(1)

    except asyncio.TimeoutError:
        logger.exception("Timed out while executing")
    except Exception:
        logger.exception("Exception while running")


def function_handler(event, context):
    return
```

```
logging.basicConfig(level=logging.INFO)
# Start up this sample code
main(logger=logging.getLogger())
```

5. Zip item berikut ke dalam file bernama `transfer_stream_python.zip`. Ini adalah paket deployment fungsi Lambda Anda.

   - transfer_stream.py. Logika aplikasi.
   - greengrasssdk. Diperlukan perpustakaan fungsi Python Greengrass Lambda yang menerbitkankan pesan MQTT.

     Operasi pengelola pengaliran tersedia dalam versi 1.5.0 atau yang lebih baru dari AWS IoT Greengrass Core SDK for Python.
   - Dependensi yang Anda instal untuk AWS IoT Greengrass Core SDK for Python (sebagai contoh, direktori `cbor2` ini).

   Ketika Anda membuat file `zip` ini, termasuk hanya item ini, bukan folder yang berisi.

Langkah 2: Buat fungsi Lambda

1. Buat IAM role sehingga Anda dapat lulus di dalam peran ARN ketika Anda membuat fungsi.

   JSON Expanded

   ```
   aws iam create-role --role-name Lambda_empty --assume-role-policy '{
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Principal": {
                   "Service": "lambda.amazonaws.com"
               },
              "Action": "sts:AssumeRole"
           }
       ]
   }'
   ```

JSON Single-line

```
aws iam create-role --role-name Lambda_empty --assume-role-policy '{"Version":
 "2012-10-17", "Statement": [{"Effect": "Allow", "Principal": {"Service":
 "lambda.amazonaws.com"},"Action": "sts:AssumeRole"}]}'
```

> **Note**
>
> AWS IoT Greengrass tidak menggunakan peran ini karena izin untuk fungsi Lambda Greengrass ditentukan di dalam peran grup Greengrass. Untuk tutorial ini, Anda membuat peran kosong.

2. Salin `Arn` dari output.
3. Gunakan AWS Lambda API untuk membuat `TransferStream` fungsi. Perintah berikut mengasumsikan bahwa file zip ada di dalam direktori saat ini.

   - Ganti *`role-arn`* dengan `Arn` yang Anda salin.

```
aws lambda create-function \
--function-name TransferStream \
--zip-file fileb://transfer_stream_python.zip \
--role role-arn \
--handler transfer_stream.function_handler \
--runtime python3.7
```

4. Mempublikasikan versi fungsi.

```
aws lambda publish-version --function-name TransferStream --description 'First
 version'
```

5. Buat alias untuk versi yang dipublikasikan.

   Grup Greengrass dapat mereferensi fungsi Lambda dengan alias (direkomendasikan) atau dengan versi. Menggunakan alias membuatnya lebih mudah untuk mengelola pembaruan kode karena Anda tidak perlu mengubah tabel langganan atau definisi grup ketika kode fungsi diperbarui. Sebaliknya, Anda hanya mengarahkan alias ke versi fungsi baru.

```
aws lambda create-alias --function-name TransferStream --name GG_TransferStream --
function-version 1
```

> **Note**
>
> AWS IoT Greengrass tidak mendukung alias Lambda untuk versi $TERBARU ini.

6. Salin `AliasArn` dari output. Anda menggunakan nilai ini ketika Anda mengonfigurasi fungsi untuk AWS IoT Greengrass.

Sekarang Anda siap untuk mengonfigurasi fungsi untuk AWS IoT Greengrass.

Langkah 3: Buat definisi fungsi dan versi

Langkah ini menciptakan versi definisi fungsi yang mereferensi fungsi Lambda `GGStreamManager` sistem dan fungsi Lambda `TransferStream` yang ditetapkan pengguna Anda. Untuk mengaktifkan pengelola pengaliran ketika Anda menggunakan AWS IoT Greengrass API, versi definisi fungsi Anda harus menyertakan fungsi `GGStreamManager` ini.

1. Buat definisi fungsi dengan versi awal yang berisi sistem dan fungsi Lambda yang ditetapkan pengguna.

   Versi definisi berikut mengizinkan pengelola pengaliran dengan default pengaturan parameter. Untuk mengonfigurasi pengaturan kustom, Anda harus menentukan variabel lingkungan untuk parameter pengelola pengaliran yang sesuai. Sebagai contoh, lihat the section called “Mengaktifkan, menonaktifkan, atau mengonfigurasi pengelola pengaliran”. AWS IoT Greengrass menggunakan pengaturan default untuk parameter yang dihilangkan. `MemorySize` setidaknya harus `128000`. `Pinned` harus diatur ke `true`.

   > **Note**
   >
   > Fungsi Lambda berumur panjang (atau disematkan) memulai secara otomatis setelah AWS IoT Greengrass dimulai dan terus berjalan dalam kontainernya sendiri. Hal ini berbeda dengan fungsi Lambda sesuai permintaan ini, yang dimulai ketika diaktifkan dan berhenti ketika tidak ada tugas yang tersisa untuk dijalankan. Untuk informasi selengkapnya, lihat the section called “Konfigurasi siklus hidup”.

- Ganti*arbitrary-function-id*dengan nama untuk fungsi, seperti**stream-manager**.
- Ganti *alias-arn* dengan `AliasArn` yang Anda salin ketika Anda membuat alias untuk `TransferStream` fungsi Lambda.

JSON expanded

```
aws greengrass create-function-definition --name MyGreengrassFunctions --
initial-version '{
    "Functions": [
        {
            "Id": "arbitrary-function-id",
            "FunctionArn": "arn:aws:lambda:::function:GGStreamManager:1",
            "FunctionConfiguration": {
                "MemorySize": 128000,
                "Pinned": true,
                "Timeout": 3
            }
        },
        {
            "Id": "TransferStreamFunction",
            "FunctionArn": "alias-arn",
            "FunctionConfiguration": {
                "Executable": "transfer_stream.function_handler",
                "MemorySize": 16000,
                "Pinned": true,
                "Timeout": 5
            }
        }
    ]
}'
```

JSON single

```
aws greengrass create-function-definition \
--name MyGreengrassFunctions \
--initial-version '{"Functions": [{"Id": "arbitrary-function-
id","FunctionArn": "arn:aws:lambda:::function:GGStreamManager:1",
 "FunctionConfiguration": {"Environment": {"Variables":
```

```
{"STREAM_MANAGER_STORE_ROOT_DIR": "/data","STREAM_MANAGER_SERVER_PORT":
 "1234","STREAM_MANAGER_EXPORTER_MAX_BANDWIDTH": "20000"}},"MemorySize":
 128000,"Pinned": true,"Timeout": 3}},{"Id": "TransferStreamFunction",
 "FunctionArn": "alias-arn", "FunctionConfiguration": {"Executable":
 "transfer_stream.function_handler", "MemorySize": 16000,"Pinned":
 true,"Timeout": 5}}]}'
```

> **Note**
>
> `Timeout` diperlukan oleh versi definisi fungsi, tetapi `GGStreamManager` tidak menggunakannya. Untuk informasi lebih lanjut tentang `Timeout` dan pengaturan tingkat grup lainnya, lihat the section called “Mengontrol eksekusi fungsi Greengrass Lambda”.

2. Salin `LatestVersionArn` dari output. Anda menggunakan nilai ini untuk menambahkan versi definisi fungsi untuk versi grup yang Anda terapkan ke inti.

Langkah 4: Buat definisi pencatat dan versi

Konfigurasi pengaturan pencatatan grup. Untuk tutorial ini, Anda mengonfigurasi AWS IoT Greengrass komponen sistem dan fungsi Lambda yang ditentukan pengguna untuk menulis log ke sistem file perangkat inti. Anda dapat menggunakan catatan untuk memecahkan masalah yang mungkin Anda alami. Untuk informasi selengkapnya, lihat the section called “Pemantauan dengan AWS IoT Greengrass log”.

1. Membuat definisi pencatat yang mencakup versi awal.

   JSON Expanded

```
aws greengrass create-logger-definition --name "LoggingConfigs" --initial-
version '{
    "Loggers": [
        {
            "Id": "1",
            "Component": "GreengrassSystem",
            "Level": "INFO",
            "Space": 10240,
            "Type": "FileSystem"
        },
        {
```

```
            "Id": "2",
            "Component": "Lambda",
            "Level": "INFO",
            "Space": 10240,
            "Type": "FileSystem"
        }
    ]
}'
```

JSON Single-line

```
aws greengrass create-logger-definition \
    --name "LoggingConfigs" \
    --initial-version '{"Loggers":
[{"Id":"1","Component":"GreengrassSystem","Level":"INFO","Space":10240,"Type":"FileSyste
{"Id":"2","Component":"Lambda","Level":"INFO","Space":10240,"Type":"FileSystem"}]}'
```

2. Salin `LatestVersionArn` dari definisi pencatat dari output. Anda menggunakan nilai ini untuk menambahkan versi definisi pencatat ke versi grup yang Anda men-deploy ke core.

Langkah 5: Dapatkan ARN versi definisi core Anda

Dapatkan ARN versi definisi core untuk ditambahkan ke versi grup baru Anda. Untuk men-deploy versi grup, hal itu harus mereferensi versi definisi core yang berisi tepat satu core.

1. Dapatkan ID dari grup Greengrass target dan versi grup. Prosedur ini mengasumsikan bahwa ini adalah versi grup dan grup terbaru. Query berikut mengembalikan grup yang paling baru dibuat.

```
aws greengrass list-groups --query "reverse(sort_by(Groups, &CreationTimestamp))
[0]"
```

Atau, Anda dapat melakukan query berdasarkan nama. Nama grup tidak perlu unik, sehingga beberapa grup mungkin ditampilkan.

```
aws greengrass list-groups --query "Groups[?Name=='MyGroup']"
```

> **Note**
>
> Anda juga dapat menemukan nilai-nilai ini di konsol AWS IoT tersebut. ID grup ditampilkan pada halaman Pengaturan grup. ID versi grup ditampilkan pada grupDeploymentTab.

2. Salin `Id` dari grup target dari output. Anda menggunakan ini untuk mendapatkan versi definisi core dan ketika Anda men-deploy grup.
3. Salin `LatestVersion` dari output, yang merupakan ID dari versi terakhir ditambahkan ke grup. Anda menggunakan ini untuk mendapatkan versi definisi core.
4. Dapatkan ARN dari versi definisi core:

   a. Dapatkan versi grup.

   - Ganti *`id-grup`* dengan `Id` yang Anda salin untuk grup.
   - Ganti*`group-version-id`*dengan`LatestVersion`yang Anda salin untuk grup.

   ```
   aws greengrass get-group-version \
   --group-id group-id \
   --group-version-id group-version-id
   ```

   b. Salin `CoreDefinitionVersionArn` dari output. Anda menggunakan nilai ini untuk menambahkan versi definisi core untuk versi grup yang Anda men-deploy ke core.

Langkah 6: Buat versi grup

Sekarang, Anda siap untuk membuat versi grup yang berisi entitas yang ingin Anda gunakan. Anda melakukannya dengan membuat versi grup yang mereferensi versi target dari setiap tipe komponen. Untuk tutorial ini, Anda menyertakan versi definisi core, versi definisi fungsi, dan versi definisi logger.

1. Buat versi grup.

   - Ganti *`grup-id`* dengan `Id` yang Anda salin untuk grup.
   - Ganti*`core-definition-version-arn`*dengan`CoreDefinitionVersionArn`yang Anda salin untuk versi definisi core.

- Ganti*function-definition-version-arn*dengan`LatestVersionArn`nyang Anda salin untuk versi definisi fungsi baru Anda.
- Ganti*logger-definition-version-arn*dengan`LatestVersionArn`nyang Anda salin untuk versi definisi logger baru Anda.

```
aws greengrass create-group-version \
--group-id group-id \
--core-definition-version-arn core-definition-version-arn \
--function-definition-version-arn function-definition-version-arn \
--logger-definition-version-arn logger-definition-version-arn
```

2. Salin `Version` dari output. Ini adalah ID dari versi grup baru.

Langkah 7: Buat deployment

Men-deploy grup ke perangkat core.

1. Pastikan bahwa core AWS IoT Greengrass sedang berjalan. Jalankan perintah berikut di terminal Raspberry Pi Anda, sesuai kebutuhan.

   a. Untuk memeriksa apakah daemon sedang berjalan:

   ```
   ps aux | grep -E 'greengrass.*daemon'
   ```

   Jika outputnya berisi entri `root` untuk `/greengrass/ggc/packages/`*ggc-version*`/bin/daemon`, maka daemon sedang berjalan.

   > **Note**
   >
   > Versi di jalur tergantung pada versi perangkat lunak AWS IoT Greengrass core yang diinstal pada perangkat core Anda.

   b. Untuk memulai daemon:

   ```
   cd /greengrass/ggc/core/
   sudo ./greengrassd start
   ```

2. Buat deployment.

- Ganti *grup-id* dengan `Id` yang Anda salin untuk grup.
- Ganti*group-version-id*dengan`Version`yang Anda salin untuk versi grup baru.

```
aws greengrass create-deployment \
--deployment-type NewDeployment \
--group-id group-id \
--group-version-id group-version-id
```

3. Salin `DeploymentId` dari output.
4. Dapatkan status deployment.

   - Ganti *grup-id* dengan `Id` yang Anda salin untuk grup.
   - Ganti *id-deployment* dengan `DeploymentId` yang Anda salin untuk deployment.

   ```
   aws greengrass get-deployment-status \
   --group-id group-id \
   --deployment-id deployment-id
   ```

   Jika statusnya `Success`, deployment berhasil. Untuk langkah-langkah penyelesaian masalah, lihat Memecahkan masalah.

Langkah 8: Uji aplikasi

Fungsi Lambda `TransferStream` menghasilkan data perangkat yang disimulasikan. Ini menulis data ke pengaliran yang diekspor oleh pengelola pengaliran ke Kinesis data stream target.

1. Di konsol Amazon Kinesis, di bawahKinesis data streams, pilihMyKinesisStream.

   > **Note**
   >
   > Jika Anda menjalankan tutorial tanpa Kinesis data streams target, periksa berkas log untuk pengelola pengaliran (`GGStreamManager`). Jika itu berisi `export stream MyKinesisStream doesn't exist` dalam pesan kesalahan, maka uji ini berhasil. Kesalahan ini berarti bahwa layanan mencoba untuk mengekspor ke pengaliran tetapi pengaliran tidak ada.

2. PadaMyKinesisStreamhalaman, pilihPemantauan. Jika uji berhasil, Anda akan melihat data di bagan Pasang Catatan ini. Tergantung pada koneksi Anda, mungkin diperlukan waktu satu menit sebelum data ditampilkan.

   > **Important**
   >
   > Setelah selesai melakukan pengujian, hapus Kinesis data stream agar tidak menimbulkan biaya tambahan.
   > Atau, jalankan perintah berikut untuk menghentikan daemon Greengrass. Hal ini mencegah core mengirim pesan hingga Anda siap untuk melanjutkan pengujian.
   >
   > ```
   > cd /greengrass/ggc/core/
   > sudo ./greengrassd stop
   > ```

3. HapusTransferStreamFungsi Lambda dari core.

   a. Ikuti the section called “Buat versi grup” untuk membuat versi grup baru. tetapi menghapus `--function-definition-version-arn` di dalam opsi perintah `create-group-version` ini. Atau, buat versi definisi fungsi yang tidak mencakupTransferStreamFungsi Lambda.

      > **Note**
      >
      > Dengan menghilangkan fungsi Lambda `GGStreamManager` sistem dari versi grup yang di-deploy, Anda menonaktifkan pengelolaan pengaliran pada core.

   b. Ikuti the section called “Buat deployment” untuk men-deploy versi grup baru.

Untuk melihat pencatatan informasi atau pemecahan masalah dengan aliran, periksa log untuk fungsi `TransferStream` dan `GGStreamManager` tersebut. Anda harus memiliki izin `root` untuk membaca catatan AWS IoT Greengrass pada sistem file.

- `TransferStream` menulis entri log ke *`greengrass-root`*`/ggc/var/log/user/`*`region`*`/`*`account-id`*`/TransferStream.log`.
- `GGStreamManager` menulis entri log ke *`greengrass-root`*`/ggc/var/log/system/GGStreamManager.log`.

Jika Anda memerlukan informasi pemecahan masalah lainnya, Anda dapat mengatur tingkat pencatatan `Lambda` ke `DEBUG` lalu membuat dan men-deploy versi grup baru.

Lihat juga

- Mengelola aliran data
- the section called “Gunakan StreamManagerClient untuk bekerja dengan aliran”
- the section called “Konfigurasi ekspor untuk tujuan AWS Cloud yang didukung”
- the section called “Konfigurasi pengelola pengaliran”
- the section called “Ekspor aliran data (konsol)”
- AWS Identity and Access Management perintah (IAM) dalam AWS CLI Referensi Perintah
- AWS Lambda perintah dalam AWS CLI Referensi Perintah
- AWS IoT Greengrass perintah dalam AWS CLI Referensi Perintah

Men-deploy rahasia ke AWS IoT Greengrass core

Fitur ini tersedia untuk AWS IoT Greengrass Core v1.7 dan yang lebih baru.

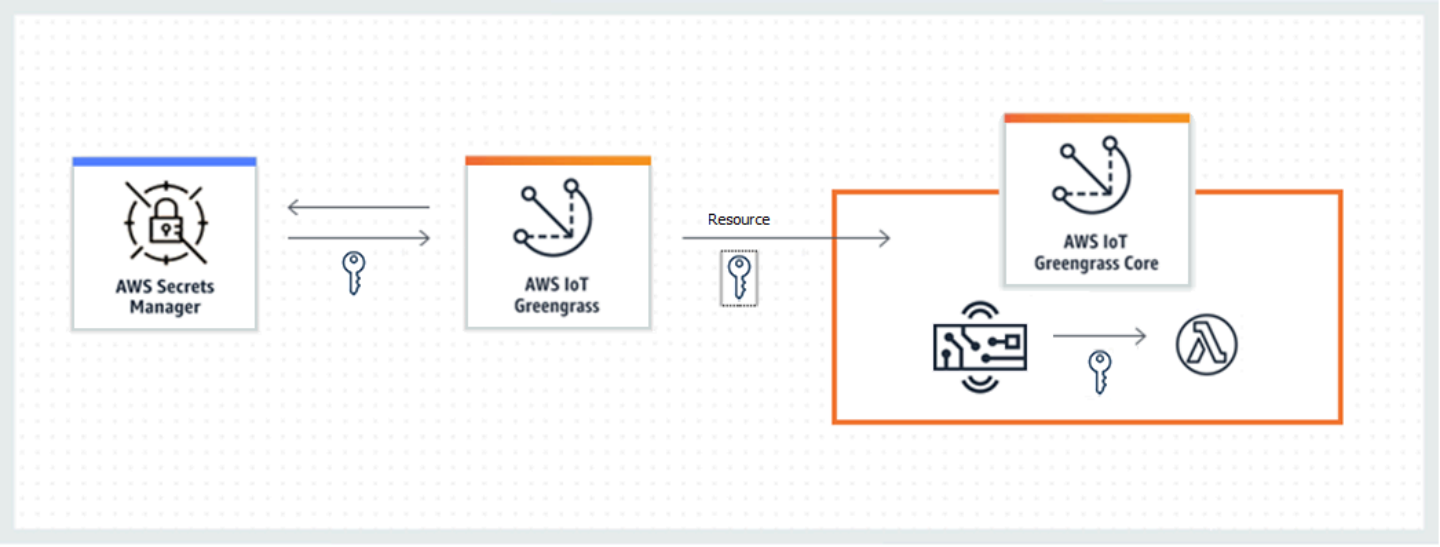
AWS IoT Greengrass memungkinkan Anda mengautentikasi dengan layanan dan aplikasi dari perangkat Greengrass tanpa kata sandi hard-coding, token, atau rahasia lainnya.

AWS Secrets Manager adalah layanan yang dapat Anda gunakan untuk menyimpan dan mengelola rahasia Anda di cloud dengan aman. AWS IoT Greengrass memperluas Secrets Manager ke perangkat core Greengrass, sehingga konektor Anda dan fungsi Lambda dapat menggunakan rahasia lokal untuk berinteraksi dengan layanan dan aplikasi. Contohnya, konektor Twilio Notifications menggunakan token autentikasi yang tersimpan secara lokal.

Untuk mengintegrasikan rahasia ke grup Greengrass, Anda membuat sumber daya grup yang mereferensi rahasia Secrets Manager. Sumber daya rahasia ini mereferensi rahasia cloud oleh ARN. Untuk mempelajari cara membuat, mengelola, dan menggunakan sumber daya rahasia, lihat the section called “Bekerja dengan sumber daya rahasia”.

AWS IoT Greengrass mengenkripsi rahasia Anda saat transit dan saat tidak digunakan. Selama deployment grup, AWS IoT Greengrass mengambil rahasia dari Secrets Manager dan menciptakan lokal, salinan terenkripsi pada core Greengrass. Setelah Anda memutar rahasia cloud Anda di Secrets Manager, redeploy grup untuk men-deploy nilai diperbarui ke core.

Diagram berikut menunjukkan proses tingkat tinggi penggelaran rahasia ke core. Rahasia dienkripsi saat transit dan saat tidak digunakan.



Menggunakan AWS IoT Greengrass untuk menyimpan rahasia Anda secara lokal memberikan keuntungan berikut:

- Dipisah dari kode (tidak hard-coded). Hal ini mendukung kredensial yang dikelola secara terpusat dan membantu melindungi data sensitif dari risiko kompromi.
- Tersedia untuk skenario offline. Konektor dan fungsi dapat mengakses layanan dan perangkat lunak lokal dengan aman ketika terputus dari internet.
- Akses yang dikendalikan ke rahasia. Hanya konektor dan fungsi terotorisasi dalam grup yang dapat mengakses rahasia Anda. AWS IoT Greengrass menggunakan enkripsi kunci privat untuk mengamankan rahasia Anda. Rahasia dienkripsi saat transit dan saat tidak digunakan. Untuk informasi selengkapnya, lihat the section called “Enkripsi rahasia”.
- Rotasi terkontrol. Setelah Anda memutar rahasia Anda di Secrets Manager, redeploy grup Greengrass untuk memperbarui salinan lokal rahasia Anda. Untuk informasi selengkapnya, lihat the section called “Membuat dan mengelola rahasia”.

> **Important**
>
> AWS IoT Greengrass tidak secara otomatis memperbarui nilai-nilai rahasia lokal setelah versi cloud diputar. Untuk memperbarui nilai-nilai lokal, Anda harus redeploy grup.

Enkripsi rahasia

AWS IoT Greengrass mengenkripsi rahasia saat transit dan saat tidak digunakan.

> **Important**
>
> Pastikan bahwa fungsi Lambda yang ditetapkan pengguna milik Anda dapat menangani rahasia dengan aman dan jangan mencatat data sensitif apa pun yang disimpan dalam rahasia. Untuk informasi lebih lanjut, lihat Mengurangi risiko Pencatatan dan Debugging fungsi Lambda Anda dalam AWS Secrets Manager Panduan Pengguna. Meskipun dokumentasi ini secara khusus mengacu pada fungsi rotasi, rekomendasi juga berlaku untuk fungsi Lambda Greengrass.

Enkripsi dalam transit

AWS IoT Greengrass menggunakan Transport Layer Security (TLS) untuk mengenkripsi semua komunikasi melalui internet dan jaringan lokal. Hal ini melindungi rahasia saat transit, yang muncul ketika rahasia diambil dari Secrets Manager dan di-deploy ke core. Untuk dukungan TLS cipher suite, lihat the section called “TLS cipher suite mendukung”.

Enkripsi saat diam

AWS IoT Greengrass menggunakan kunci privat yang ditentukan dalam `config.json` untuk enkripsi rahasia yang disimpan pada core. Untuk alasan ini, penyimpanan yang aman dengan kunci privat sangat penting untuk melindungi rahasia lokal. Di AWS model tanggung jawab bersama, merupakan tanggung jawab pelanggan untuk menjamin penyimpanan kunci privat yang aman pada perangkat core.

AWS IoT Greengrass mendukung dua mode penyimpanan kunci privat:

- Menggunakan modul keamanan perangkat keras. Untuk informasi selengkapnya, lihat the section called “Integrasi keamanan perangkat keras”.

  > **Note**
  >
  > Saat ini, AWS IoT Greengrass hanya mendukung mekanisme padding PKCS #1 v1.5 untuk enkripsi dan dekripsi rahasia lokal ketika menggunakan kunci privat berbasis perangkat keras. Jika Anda mengikuti petunjuk yang disediakan vendor untuk secara manual menghasilkan kunci privat berbasis perangkat keras, pastikan untuk memilih PKCS #1 v1.5. AWS IoT Greengrass tidak mendukung Optimal Asymmetric Encryption Padding (OAEP).

- Menggunakan izin sistem file (default).

Kunci privat digunakan untuk mengamankan kunci data, yang digunakan untuk mengenkripsi rahasia lokal. Kunci data diputar dengan setiap deployment grup.

Core AWS IoT Greengrass adalah satu-satunya entitas yang memiliki akses ke kunci privat. Konektor Greengrass atau fungsi Lambda yang berafiliasi dengan sumber daya rahasia mendapatkan nilai rahasia dari core.

Persyaratan

Ini adalah persyaratan untuk dukungan rahasia lokal:

- Anda harus menggunakan AWS IoT Greengrass Core v1.7 atau yang lebih baru.
- Untuk mendapatkan nilai-nilai rahasia lokal, fungsi Lambda yang ditetapkan pengguna milik Anda harus menggunakan AWS IoT Greengrass Core SDK v1.3.0 atau yang lebih baru.
- Kunci privat yang digunakan untuk enkripsi rahasia lokal harus ditentukan dalam file konfigurasi Greengrass. Secara default, AWS IoT Greengrass menggunakan kunci privat core yang disimpan dalam sistem file. Untuk menyediakan kunci privat milik Anda sendiri, lihat the section called "Tentukan kunci privat untuk enkripsi rahasia". Hanya jenis kunci RSA yang didukung.

  > **Note**
  >
  > Saat ini, AWS IoT Greengrass hanya mendukung mekanisme padding PKCS #1 v1.5 untuk enkripsi dan dekripsi rahasia lokal ketika menggunakan kunci privat berbasis perangkat keras. Jika Anda mengikuti petunjuk yang disediakan vendor untuk secara manual menghasilkan kunci privat berbasis perangkat keras, pastikan untuk memilih PKCS #1 v1.5. AWS IoT Greengrass tidak mendukung Optimal Asymmetric Encryption Padding (OAEP).

- AWS IoT Greengrass harus diberikan izin untuk mendapatkan nilai-nilai rahasia Anda. Hal ini memungkinkan AWS IoT Greengrass untuk mengambil nilai-nilai selama deployment grup. Jika Anda menggunakan peran layanan default Greengrass, maka AWS IoT Greengrass memiliki akses ke rahasia pada nama yang dimulai dengan greengrass-. Untuk menyesuaikan akses, lihat the section called "Mengizinkan AWS IoT Greengrass untuk mendapatkan nilai rahasia".

  > **Note**
  >
  > Kami menyarankan agar Anda menggunakan konvensi penamaan ini untuk mengidentifikasi rahasia yang diizinkan AWS IoT Greengrass untuk mengakses, bahkan jika Anda menyesuaikan izin. Konsol tersebut menggunakan izin yang berbeda untuk membaca rahasia Anda, jadi Anda mungkin dapat memilih rahasia di konsol tersebut di mana AWS IoT Greengrass tidak memiliki izin untuk mengambil. Menggunakan konvensi penamaan dapat membantu menghindari konflik izin, yang mengakibatkan error deployment.

Tentukan kunci privat untuk enkripsi rahasia

Dalam prosedur ini, Anda menyediakan jalur ke kunci privat yang digunakan untuk enkripsi rahasia lokal. Hal ini harus menjadi kunci RSA dengan panjang minimum 2048 bit. Untuk informasi lebih lanjut tentang kunci privat yang digunakan pada core AWS IoT Greengrass ini, lihat the section called "Keamanan utama".

AWS IoT Greengrass mendukung dua mode penyimpanan kunci privat: berbasis perangkat keras atau berbasis sistem file (default). Untuk informasi selengkapnya, lihat the section called "Enkripsi rahasia".

Ikuti prosedur ini hanya jika Anda ingin mengubah konfigurasi default, yang menggunakan kunci privat core dalam sistem file. Langkah-langkah ini ditulis dengan asumsi bahwa Anda membuat grup dan core seperti yang dijelaskan di Modul 2 dari tutorial Memulai Dengan.

1. Buka file `config.json` yang terletak di direktori `/greengrass-root/config` ini.

   > **Note**
   >
   > *greengrass-root* mewakili jalur di mana perangkat lunak AWS IoT Greengrass core diinstal pada perangkat Anda. Biasanya, adalah direktori `/greengrass` ini.

2. Di objek `crypto.principals.SecretsManager` tersebut, untuk properti `privateKeyPath` ini, masukkan jalur dari kunci privat:

   - Jika kunci privat Anda disimpan dalam sistem file, tentukan jalur absolut ke kunci. Misalnya:

     ```
     "SecretsManager" : {
       "privateKeyPath" : "file:///somepath/hash.private.key"
     }
     ```

   - Jika kunci privat Anda disimpan dalam modul keamanan perangkat keras (HSM), tentukan jalur menggunakan skema URI RFC 7512 PKC #11 ini. Misalnya:

     ```
     "SecretsManager" : {
       "privateKeyPath" : "pkcs11:object=private-key-label;type=private"
     }
     ```

     Untuk informasi selengkapnya, lihat the section called "Konfigurasi keamanan perangkat keras".

> **Note**
>
> Saat ini, AWS IoT Greengrass hanya mendukung mekanisme padding PKCS #1 v1.5 untuk enkripsi dan dekripsi rahasia lokal ketika menggunakan kunci privat berbasis perangkat keras. Jika Anda mengikuti petunjuk yang disediakan vendor untuk secara manual menghasilkan kunci privat berbasis perangkat keras, pastikan untuk memilih PKCS #1 v1.5. AWS IoT Greengrass tidak mendukung Optimal Asymmetric Encryption Padding (OAEP).

Mengizinkan AWS IoT Greengrass untuk mendapatkan nilai rahasia

Dalam prosedur ini, Anda menambahkan kebijakan inline agar peran layanan Greengrass memungkinkan AWS IoT Greengrass untuk mendapatkan nilai-nilai rahasia Anda.

Ikuti prosedur ini hanya jika Anda ingin memberikan AWS IoT Greengrass izin kustom untuk rahasia Anda atau jika peran layanan Greengrass Anda tidak termasuk Kebijakan terkelola `AWSGreengrassResourceAccessRolePolicy` ini. `AWSGreengrassResourceAccessRolePolicy` memberikan akses ke rahasia pada nama yang dimulai dengan greengrass-.

1. Jalankan perintah CLI berikut untuk mendapatkan ARN dari peran layanan Greengrass:

   ```
   aws greengrass get-service-role-for-account --region region
   ```

   ARN yang kembali berisi nama peran.

   ```
   {
     "AssociatedAt": "time-stamp",
     "RoleArn": "arn:aws:iam::account-id:role/service-role/role-name"
   }
   ```

   Anda menggunakan ARN atau nama pada langkah berikut.

2. Menambahkan kebijakan inline yang mengizinkan tindakan `secretsmanager:GetSecretValue` ini. Untuk instruksi, lihat Menambahkan dan menghapus kebijakan IAM di Panduan Pengguna IAM.

   Anda dapat memberikan akses terperinci dengan daftar rahasia secara eksplisit atau menggunakan wildcard skema penamaan * ini, atau Anda dapat memberikan akses bersyarat ke versi atau tag rahasia. Misalnya, kebijakan berikut mengizinkan AWS IoT Greengrass untuk membaca hanya rahasia yang ditentukan.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": [
                "arn:aws:secretsmanager:region:account-id:secret:greengrass-
SecretA-abc",
                "arn:aws:secretsmanager:region:account-id:secret:greengrass-
SecretB-xyz"
            ]
        }
    ]
}
```

   > **Note**
   >
   > Jika Anda menggunakan kunci AWS KMS yang dikelola pelanggan untuk mengenkripsi rahasia, peran layanan Greengrass juga harus mengizinkan tindakan `kms:Decrypt` ini.

Untuk informasi lebih lanjut tentang kebijakan IAM untuk Secrets Manager, lihat Autentikasi dan kontrol akses untuk AWS Secrets Manager dan Tindakan, sumber daya, dan kunci konteks yang dapat Anda gunakan dalam kebijakan IAM atau kebijakan rahasia untuk AWS Secrets Manager di AWS Secrets Manager Panduan Pengguna.

Lihat juga

- Apa itu AWS Secrets Manager? dalam AWS Secrets Manager Panduan Penggguna
- PKCS #1: Enkripsi RSA Versi 1.5

Bekerja dengan sumber daya rahasia

AWS IoT Greengrass menggunakan Sumber daya rahasia untuk mengintegrasikan rahasia dari AWS Secrets Manager ke dalam grup Greengrass. Sumber daya rahasia ini mereferensi rahasia Secrets Manager. Untuk informasi lebih lanjut, lihat Men-deploy rahasia ke core.

Pada perangkat core AWS IoT Greengrass tersebut, konektor dan fungsi Lambda dapat menggunakan sumber daya rahasia untuk mengautentikasi dengan layanan dan aplikasi, tanpa kata sandi hard-coding, token, atau kredensial lainnya.

Membuat dan mengelola rahasia

Dalam grup Greengrass, sumber daya rahasia mereferensi ARN dari rahasia Secrets Manager. Ketika sumber rahasia di-deploy ke core, nilai rahasia dienkripsi dan tersedia untuk konektor berafiliasi dan fungsi Lambda. Untuk informasi selengkapnya, lihat the section called “Enkripsi rahasia”.

Anda menggunakan Secrets Manager untuk membuat dan mengelola versi cloud rahasia Anda. Anda menggunakan AWS IoT Greengrass untuk membuat, mengelola, dan menerapkan sumber daya rahasia Anda.

> **Important**
>
> Kami menyarankan agar Anda mengikuti praktik terbaik untuk memutar rahasia Anda di Secrets Manager. Kemudian, men-deploy grup Greengrass untuk memperbarui salinan lokal rahasia Anda. Untuk informasi lebih lanjut, lihat Memutar rahasia AWS Secrets Manager Anda di AWS Secrets Manager Panduan Pengguna.

Untuk membuat rahasia tersedia di inti Greengrass

1. Buat rahasia di Secrets Manager. Ini adalah versi cloud dari rahasia Anda, yang disimpan secara terpusat dan dikelola di Secrets Manager. Tugas manajemen termasuk memutar nilai-nilai rahasia dan menerapkan kebijakan sumber daya.
2. Buat sumber daya rahasia di AWS IoT Greengrass. Ini adalah jenis sumber daya grup yang mereferensi rahasia cloud oleh ARN. Anda dapat mereferensikan rahasia hanya sekali per grup.
3. Konfigurasi konektor atau fungsi Lambda Anda. Anda harus mengafiliasi sumber daya dengan konektor atau fungsi dengan menentukan parameter atau properti yang sesuai. Hal ini mengizinkan mereka untuk mendapatkan nilai sumber daya rahasia yang di-deploy secara lokal. Untuk informasi lebih lanjut, lihat the section called “Menggunakan rahasia lokal”.
4. Men-deploy grup Greengrass. Selama deployment, AWS IoT Greengrass mengambil nilai rahasia cloud dan menciptakan (atau membarui) rahasia lokal pada core.

Secrets Manager mencatat acara di AWS CloudTrail setiap kali AWS IoT Greengrass mengambil nilai rahasia. AWS IoT Greengrass tidak mencatat acara apa pun yang terkait dengan deployment atau penggunaan rahasia lokal. Untuk informasi lebih lanjut tentang pencatatan Secrets Manager, lihat Pantau penggunaan rahasia AWS Secrets Manager Anda di AWS Secrets Manager Panduan Pengguna.

Termasuk label staging dalam sumber daya rahasia

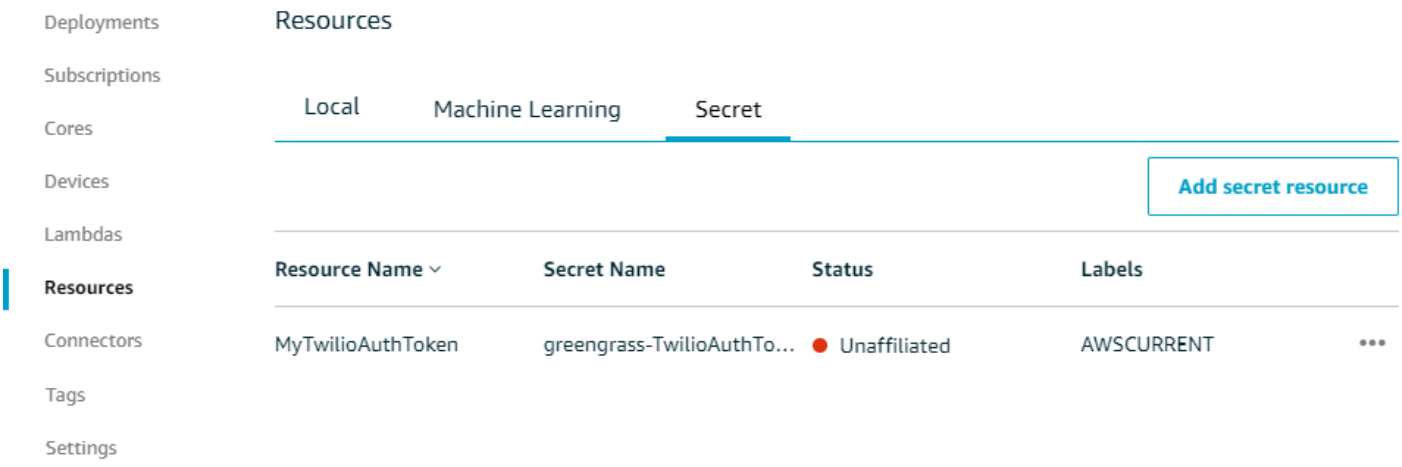
Secrets Manager menggunakan label staging untuk mengidentifikasi versi tertentu dari nilai rahasia. Label staging dapat ditetapkan sistem atau ditetapkan pengguna. Secrets Manager menugaskan label `AWSCURRENT` ke versi terbaru dari nilai rahasia. Label staging umumnya digunakan untuk mengelola rotasi rahasia. Untuk informasi lebih lanjut tentang pembuatan versioning Secrets Manager, lihat Istilah dan konsep kunci untuk AWS Secrets Manager di AWS Secrets Manager Panduan Pengguna.

Sumber daya rahasia selalu menyertakan label staging `AWSCURRENT` ini, dan mereka secara opsional dapat menyertakan label staging lain jika mereka diperlukan oleh fungsi Lambda atau konektor. Selama deployment grup, AWS IoT Greengrass mengambil nilai-nilai label staging yang direferensikan dalam grup, dan kemudian menciptakan atau memperbarui nilai-nilai yang sesuai pada core.

Membuat dan mengelola sumber daya rahasia (konsol)

Membuat sumber daya rahasia (konsol)

Di konsol AWS IoT Greengrass tersebut, Anda membuat dan mengelola sumber daya rahasia dari tab Rahasia di halaman grup Sumber Daya ini. Untuk tutorial yang membuat sumber daya rahasia dan menambahkannya ke grup, lihat the section called “Cara membuat sumber daya rahasia (konsol)” dan the section called “Memulai dengan konektor (konsol)”.



> **Note**
>
> Sebagai alternatif, konsol tersebut mengizinkan Anda membuat sumber rahasia dan rahasia saat Anda mengonfigurasi konektor atau fungsi Lambda. Anda dapat melakukan hal ini dari konektor halaman Mengonfigurasi parameter atau fungsi Lambda halaman Sumber Daya ini.

Mengelola sumber daya rahasia (konsol)

Tugas manajemen untuk sumber daya rahasia dalam grup Greengrass Anda termasuk menambahkan sumber daya rahasia ke grup, menghapus sumber daya rahasia dari grup, dan mengubah aturan dari Label staging yang termasuk dalam sumber daya rahasia.

Jika Anda menunjuk ke rahasia yang berbeda dari Secrets Manager, Anda juga harus mengedit konektor yang menggunakan rahasia:

1. Di halaman Konfigurasi grup, pilih Konektor.
2. Dari menu kontekstual konektor, pilih Edit.

3. Halaman Edit parameter menampilkan pesan untuk memberitahu Anda bahwa ARN rahasia berubah. Untuk mengonfirmasi perubahan, pulih Simpan.

Jika Anda menghapus rahasia di Secrets Manager, maka menghapus sumber rahasia yang sesuai dari grup dan dari konektor dan fungsi Lambda yang mereferensinya. Jika tidak, selama deployment grup, AWS IoT Greengrass mengembalikan error bahwa rahasia tidak dapat ditemukan. Juga memperbarui kode fungsi Lambda Anda sesuai kebutuhan.

Membuat dan mengelola sumber daya rahasia (CLI)

Membuat sumber daya rahasia (CLI)

Di API AWS IoT Greengrass ini, rahasia adalah jenis sumber daya grup. Contoh berikut membuat definisi sumber daya dengan versi awal yang mencakup sumber daya rahasia bernama `MySecretResource`. Untuk tutorial yang membuat sumber daya rahasia dan menambahkannya ke versi grup, lihat the section called “Memulai dengan konektor (CLI)”.

Sumber daya rahasia mereferensi ARN dari Secrets Manager yang sesuai rahasia dan mencakup dua label staging selain `AWSCURRENT`, yang selalu disertakan.

```
aws greengrass create-resource-definition --name MyGreengrassResources --initial-
version '{
    "Resources": [
        {
            "Id": "my-resource-id",
            "Name": "MySecretResource",
            "ResourceDataContainer": {
                "SecretsManagerSecretResourceData": {
                    "ARN": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:greengrass-SomeSecret-KUj89s",
                    "AdditionalStagingLabelsToDownload": [
                        "Label1",
                        "Label2"
                    ]
                }
            }
        }
    ]
}'
```

Mengelola sumber daya rahasia (CLI)

Tugas manajemen untuk sumber daya rahasia dalam grup Greengrass Anda termasuk menambahkan sumber daya rahasia ke grup, menghapus sumber daya rahasia dari grup, dan mengubah aturan dari Label staging yang termasuk dalam sumber daya rahasia.

Di API AWS IoT Greengrass ini, perubahan ini dilaksanakan dengan menggunakan versi.

API AWS IoT Greengrass menggunakan versi untuk mengelola grup. Versi tidak dapat diubah, jadi untuk menambah atau mengubah komponen grup — misalnya, perangkat klien grup, fungsi, dan sumber daya — Anda harus membuat versi komponen baru atau yang diperbarui. Kemudian, Anda membuat dan men-deploy versi grup yang berisi versi target masing-masing komponen. Untuk mempelajari tentang grup, lihat the section called “AWS IoT Greengrass Grup”.

Misalnya, untuk mengubah aturan label staging untuk sumber daya rahasia:

1. Buat versi definisi sumber daya yang berisi sumber daya rahasia diperbarui. Contoh berikut menambahkan label staging ketiga untuk sumber daya rahasia dari bagian sebelumnya.

> **Note**
>
> Untuk menambahkan lebih banyak sumber daya dari versi, termasuk mereka di dalam array `Resources` ini.

```
aws greengrass create-resource-definition --name MyGreengrassResources --initial-
version '{
    "Resources": [
        {
            "Id": "my-resource-id",
            "Name": "MySecretResource",
            "ResourceDataContainer": {
                "SecretsManagerSecretResourceData": {
                    "ARN": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:greengrass-SomeSecret-KUj89s",
                    "AdditionalStagingLabelsToDownload": [
                        "Label1",
                        "Label2",
                        "Label3"
                    ]
                }
```

```
            }
        }
    ]
}'
```

2. Jika ID dari sumber daya rahasia diubah, perbarui konektor dan fungsi yang menggunakan sumber daya rahasia. Dalam versi baru, perbarui parameter atau properti yang sesuai dengan ID sumber daya. Jika ARN rahasianya berubah, Anda juga harus memperbarui parameter yang sesuai untuk konektor yang menggunakan rahasianya.

   > **Note**
   >
   > ID sumber daya adalah pengenal arbitrer yang disediakan oleh pelanggan.

3. Membuat versi grup yang berisi versi target dari setiap komponen yang ingin Anda kirim ke core.
4. Men-deploy versi grup.

Untuk tutorial yang menunjukkan cara membuat dan men-deploy sumber daya rahasia, konektor, dan fungsi, lihat the section called “Memulai dengan konektor (CLI)”.

Jika Anda menghapus rahasia di Secrets Manager, maka menghapus sumber rahasia yang sesuai dari grup dan dari konektor dan fungsi Lambda yang mereferensinya. Jika tidak, selama deployment grup, AWS IoT Greengrass mengembalikan error bahwa rahasia tidak dapat ditemukan. Juga memperbarui kode fungsi Lambda Anda sesuai kebutuhan. Anda dapat menghapus rahasia lokal dengan men-deploy versi definisi sumber daya yang tidak berisi sumber daya rahasia yang sesuai.

Menggunakan rahasia lokal di konektor dan fungsi Lambda

Konektor Greengrass dan fungsi Lambda menggunakan rahasia lokal untuk berinteraksi dengan layanan dan aplikasi. Nilai `AWSCURRENT` digunakan secara default, tetapi nilai-nilai lainnya dari Label staging termasuk sumber daya rahasia juga tersedia.

Konektor dan fungsi harus dikonfigurasi sebelum mereka dapat mengakses rahasia lokal. Hal ini mengafiliasi sumber daya rahasia dengan konektor atau fungsi.

Konektor

Jika konektor memerlukan akses ke rahasia lokal, maka sediakan parameter yang Anda konfigurasikan dengan informasi yang dibutuhkan untuk mengakses rahasia.

- Untuk mempelajari cara melakukannya di konsol AWS IoT Greengrass tersebut, lihat the section called “Memulai dengan konektor (konsol)”.
- Untuk mempelajari cara melakukannya dengan CLI AWS IoT Greengrass tersebut, lihat the section called “Memulai dengan konektor (CLI)”.

Untuk informasi tentang persyaratan bagi masing-masing konektor, lihat the section called “AWS-disediakan konektor Greengrass”.

Logika untuk mengakses dan menggunakan rahasia dibangun ke dalam konektor.

Fungsi Lambda

Untuk memungkinkan fungsi Greengrass Lambda agar mengakses rahasia lokal, Anda mengonfigurasi properti fungsi.

- Untuk mempelajari cara melakukannya di konsol AWS IoT Greengrass tersebut, lihat the section called “Cara membuat sumber daya rahasia (konsol)”.
- Untuk melakukannya di bagian API AWS IoT Greengrass tersebut, Anda memberikan informasi berikut di properti `ResourceAccessPolicies` tersebut.
  - `ResourceId`: ID sumber daya rahasia dalam grup Greengrass. Ini adalah sumber daya yang mereferensi ARN dari Secrets Manager yang sesuai rahasia.
  - `Permission`: Jenis akses yang memiliki fungsi ke sumber daya. Hanya izin `ro` (hanya baca) didukung untuk sumber daya rahasia.

Contoh berikut membuat fungsi Lambda yang dapat mengakses sumber daya rahasia `MyApiKey` tersebut.

```
aws greengrass create-function-definition --name MyGreengrassFunctions --initial-
version '{
    "Functions": [
        {
            "Id": "MyLambdaFunction",
            "FunctionArn": "arn:aws:lambda:us-
west-2:123456789012:function:myFunction:1",
            "FunctionConfiguration": {
                "Pinned": false,
                "MemorySize": 16384,
                "Timeout": 10,
                "Environment": {
                    "ResourceAccessPolicies": [
                        {
```

```
                            "ResourceId": "MyApiKey",
                            "Permission": "ro"
                        }
                    ],
                    "AccessSysfs": true
                }
            }
        }
    ]
}'
```

Untuk mengakses rahasia lokal saat waktu aktif, fungsi Lambda Greengrass memanggil fungsi `get_secret_value` dari klien `secretsmanager` di AWS IoT Greengrass Core SDK (v1.3.0 atau yang lebih baru).

Contoh berikut menunjukkan cara menggunakan AWS IoT Greengrass Core SDK for Python untuk mendapatkan rahasia. Itu melewati nama rahasia ke fungsi `get_secret_value` tersebut. `SecretId` bisa menjadi nama atau ARN dari rahasia Secrets Manager (bukan sumber daya rahasia).

```
import greengrasssdk

secrets_client = greengrasssdk.client("secretsmanager")
secret_name = "greengrass-MySecret-abc"


def function_handler(event, context):
    response = secrets_client.get_secret_value(SecretId=secret_name)
    secret = response.get("SecretString")
```

Untuk rahasia jenis teks, `get_secret_value` fungsi kembali ke string. Untuk rahasia jenis biner, maka mengembalikan string base64 yang dikodekan.

> **Important**
>
> Pastikan bahwa fungsi Lambda yang ditetapkan pengguna milik Anda dapat menangani rahasia dengan aman dan jangan mencatat data sensitif apa pun yang disimpan

dalam rahasia. Untuk informasi lebih lanjut, lihat Mengurangi risiko Pencatatan dan Debugging fungsi Lambda Anda dalam AWS Secrets Manager Panduan Pengguna. Meskipun dokumentasi ini secara khusus mengacu pada fungsi rotasi, rekomendasi juga berlaku untuk fungsi Lambda Greengrass.

Nilai rahasia saat ini dikembalikan ke default. Ini adalah versi di mana label staging `AWSCURRENT` melekat. Untuk mengakses versi yang berbeda, berikan nama label staging yang sesuai untuk argumen `VersionStage` yang opsional. Misalnya:

```
import greengrasssdk

secrets_client = greengrasssdk.client("secretsmanager")
secret_name = "greengrass-TestSecret"
secret_version = "MyTargetLabel"


# Get the value of a specific secret version
def function_handler(event, context):
    response = secrets_client.get_secret_value(
        SecretId=secret_name, VersionStage=secret_version
    )
    secret = response.get("SecretString")
```

Untuk fungsi contoh lain yang memanggil `get_secret_value`, lihat Buat paket deployment fungsi Lambda.

Cara membuat sumber daya rahasia (konsol)

Fitur ini tersedia untuk AWS IoT Greengrass Core v1.7 dan yang lebih baru.

Tutorial ini menunjukkan cara menggunakan AWS Management Console untuk menambahkan Sumber daya rahasia ke grup Greengrass. Sumber daya rahasia adalah referensi untuk sebuah rahasia dari AWS Secrets Manager. Untuk informasi selengkapnya, lihat Men-deploy rahasia ke core.

Pada perangkat core AWS IoT Greengrass tersebut, konektor dan fungsi Lambda dapat menggunakan sumber daya rahasia untuk mengautentikasi dengan layanan dan aplikasi, tanpa kata sandi hard-coding, token, atau kredensial lainnya.

Dalam tutorial ini, Anda mulai dengan membuat rahasia di konsol AWS Secrets Manager tersebut. Kemudian, di konsol AWS IoT Greengrass tersebut, Anda menambahkan sumber daya rahasia untuk grup Greengrass dari grup halaman Sumber Daya ini. Sumber daya rahasia ini mereferensi rahasia Secrets Manager. Kemudian, Anda melampirkan sumber rahasia ke fungsi Lambda, yang memungkinkan fungsi untuk mendapatkan nilai dari rahasia lokal.

> **Note**
>
> Sebagai alternatif, konsol memungkinkan Anda membuat sumber rahasia dan rahasia saat Anda mengonfigurasi konektor atau fungsi Lambda. Anda dapat melakukan hal ini dari konektor halaman Mengonfigurasi parameter atau halaman fungsi Lambda Sumber Daya tersebut.
>
> Hanya konektor yang berisi parameter rahasia yang bisa mengakses rahasia. Untuk tutorial yang menunjukkan bagaimana konektor Twilio Notifications menggunakan token autentikasi yang tersimpan secara lokal, lihat the section called “Memulai dengan konektor (konsol)”.

Tutorial ini berisi langkah-langkah tingkat tinggi berikut:

1. Buat rahasia Secrets Manager
2. Menambahkan sumber daya rahasia ke grup
3. Buat paket deployment fungsi Lambda
4. Buat fungsi Lambda
5. Menambahkan fungsi ke grup
6. Lampirkan sumber daya rahasia ke fungsi Lambda
7. Menambahkan langganan ke grup
8. Men-deploy grup
9. the section called “Tes fungsi Lambda”

Tutorial akan memakan waktu sekitar 20 menit untuk menyelesaikannya.

Prasyarat

Untuk menyelesaikan tutorial ini, Anda memerlukan:

- Sebuah grup Greengrass dan core Greengrass (v1.7 atau yang lebih baru). Untuk mempelajari cara membuat grup Greengrass dan core, lihat Memulai dengan AWS IoT Greengrass. Tutorial Memulai Dengan juga mencakup langkah-langkah untuk menginstal perangkat lunak AWS IoT Greengrass Core.
- AWS IoT Greengrass harus dikonfigurasi untuk support rahasia lokal. Untuk informasi lebih lanjut, lihat Persyaratan Rahasia.

  > **Note**
  >
  > Persyaratan ini mencakup akses ke rahasia Secrets Manager Anda. Jika Anda menggunakan peran layanan default Greengrass, Greengrass memiliki izin untuk mendapatkan nilai-nilai rahasia pada nama yang dimulai dengan Greengrass-.

- Untuk mendapatkan nilai-nilai rahasia lokal, fungsi Lambda yang ditetapkan pengguna milik Anda harus menggunakan AWS IoT Greengrass core SDK v1.3.0 atau yang lebih baru.

Langkah 1: Buat rahasia Secrets Manager

Pada langkah ini, Anda menggunakan konsol AWS Secrets Manager untuk membuat rahasia.

1. Masuk ke AWS Secrets Manager konsol.

   > **Note**
   >
   > Untuk informasi lebih lanjut tentang proses ini, lihat Langkah 1: Buat dan simpan rahasia Anda di AWS Secrets Manager dalam AWS Secrets Manager Panduan Pengguna.

2. Pilih Simpan rahasia baru.
3. Di bawah Pilih tipe rahasia, pilih Jenis rahasia lainnya.
4. Di bawah Tentukan pasangan nilai kunci yang akan disimpan untuk rahasia ini:
   - Untuk Kunci, masukkan **`test`**.
   - Untuk Nilai, masukkan **`abcdefghi`**.

5. Tetap pilih aws/secretsmanager untuk kunci enkripsi, lalu pilih Berikutnya.

> **Note**
>
> Anda tidak dikenakan biaya oleh AWS KMS jika Anda menggunakan kunci mengelola AWS default yang dibuat oleh Secrets Manager di akun Anda.

6. Untuk Nama rahasia, masukkan **`greengrass-TestSecret`**, dan pilih Selanjutnya.

> **Note**
>
> Secara default, peran layanan Greengrass memungkinkan AWS IoT Greengrass untuk mendapatkan nilai rahasia dengan nama yang dimulai dengan Greengrass-. Untuk informasi lebih lanjut, lihat persyaratan rahasia.

7. Tutorial ini tidak memerlukan rotasi, jadi pilih nonaktifkan rotasi otomatis, lalu pilih Berikutnya.
8. Pada halaman Tinjauan tersebut, tinjau pengaturan Anda, dan kemudian pilih Menyimpan.

Selanjutnya, Anda membuat sumber daya rahasia dalam grup Greengrass Anda yang mereferensi rahasia.

Langkah 2: Menambahkan sumber daya rahasia ke grup Greengrass

Pada langkah ini, Anda mengonfigurasi sumber daya grup yang mereferensi rahasia Secrets Manager.

1. Di panel navigasi AWS IoT konsol, di bawah Kelola, perluas perangkat Greengrass, lalu pilih Grup (V1).
2. Pilih grup yang ingin Anda tambahkan sumber daya rahasia.
3. Pada halaman konfigurasi grup, pilih tab Sumber Daya, lalu gulir ke bawah ke bagian Rahasia. Bagian Rahasia menampilkan sumber daya rahasia milik grup. Anda dapat menambahkan, mengedit, dan menghapus sumber daya rahasia dari bagian ini.

> **Note**
>
> Sebagai alternatif, konsol tersebut mengizinkan Anda membuat sumber rahasia dan rahasia saat Anda mengonfigurasi konektor atau fungsi Lambda. Anda dapat melakukan

hal ini dari konektor halaman Mengonfigurasi parameter atau fungsi Lambda halaman Sumber Daya ini.

4. Pilih Tambah di bawah bagian Rahasia.
5. Pada halaman Tambahkan sumber daya rahasia, **`MyTestSecret`** masukkan nama Sumber Daya.
6. Di bawah Rahasia, pilih greengrass-. TestSecret
7. Di bagian Pilih label (Opsional), label AWSCURRENT pementasan mewakili versi terbaru dari rahasia. Label ini selalu disertakan dalam sumber rahasia.

> **Note**
>
> Tutorial ini hanya membutuhkan AWSCURRENT label. Anda dapat secara opsional menyertakan label yang diperlukan oleh fungsi Lambda atau konektor.

8. Pilih Tambahkan sumber daya.

Langkah 3: Buat paket deployment fungsi Lambda

Untuk membuat fungsi Lambda, Anda harus terlebih dahulu membuat fungsi Lambda paket deployment yang berisi kode fungsi dan dependensi. Fungsi Greengrass Lambda membutuhkan AWS IoT Greengrass Core SDK untuk tugas seperti berkomunikasi dengan pesan MQTT di lingkungan core dan mengakses rahasia lokal. Tutorial ini membuat fungsi Python, sehingga Anda menggunakan versi Python dari SDK dalam paket deployment.

> **Note**
>
> Untuk mendapatkan nilai-nilai rahasia lokal, fungsi Lambda yang ditetapkan pengguna milik Anda harus menggunakan AWS IoT Greengrass core SDK v1.3.0 atau yang lebih baru.

1. Dari halaman unduh AWS IoT Greengrass Core SDK tersebut, unduh AWS IoT Greengrass Core SDK for Python ke komputer Anda.
2. Unzip paket yang diunduh untuk mendapatkan SDK. SDK adalah folder `greengrasssdk` tersebut.
3. Simpan fungsi kode Python berikut dalam sebuah file lokal bernama `secret_test.py`.

```
import greengrasssdk

secrets_client = greengrasssdk.client("secretsmanager")
iot_client = greengrasssdk.client("iot-data")
secret_name = "greengrass-TestSecret"
send_topic = "secrets/output"


def function_handler(event, context):
    """
    Gets a secret and publishes a message to indicate whether the secret was
    successfully retrieved.
    """
    response = secrets_client.get_secret_value(SecretId=secret_name)
    secret_value = response.get("SecretString")
    message = (
        f"Failed to retrieve secret {secret_name}."
        if secret_value is None
        else f"Successfully retrieved secret {secret_name}."
    )
    iot_client.publish(topic=send_topic, payload=message)
    print("Published: " + message)
```

Fungsi `get_secret_value` support nama atau ARN dari rahasia Secrets Manager untuk nilai `SecretId` tersebut. Contoh ini menggunakan nama rahasia. Untuk contoh rahasia ini, AWS IoT Greengrass mengembalikan pasangan nilai kunci: `{"test":"abcdefghi"}`.

> **Important**
>
> Pastikan bahwa fungsi Lambda yang ditetapkan pengguna milik Anda dapat menangani rahasia dengan aman dan jangan mencatat data sensitif apa pun yang disimpan dalam rahasia. Untuk informasi lebih lanjut, lihat Mengurangi risiko Pencatatan dan Debugging fungsi Lambda Anda dalam AWS Secrets Manager Panduan Pengguna. Meskipun dokumentasi ini secara khusus mengacu pada fungsi rotasi, rekomendasi juga berlaku untuk fungsi Greengrass Lambda.

4. Zip item berikut ke dalam file bernama `secret_test_python.zip`. Ketika Anda membuat file ZIP, termasuk hanya kode dan dependensi, bukan folder yang berisi.

- secret_test.py. Logika aplikasi.
- greengrasssdk. Diperlukan perpustakaan untuk semua fungsi Python Greengrass Lambda.

Ini adalah paket deployment fungsi Lambda Anda.

Langkah 4: Buat fungsi Lambda

Pada langkah ini, Anda menggunakan konsol AWS Lambda untuk membuat fungsi Lambda dan mengonfigurasinya agar menggunakan paket deployment Anda. Kemudian, Anda mempublikasikan versi fungsi dan membuat alias.

1. Pertama, buat fungsi Lambda.

   a. Di AWS Management Console, pilih Layanan, dan buka konsol AWS Lambda tersebut.

   b. Pilih Buat fungsi dan kemudian Tulis dari awal.

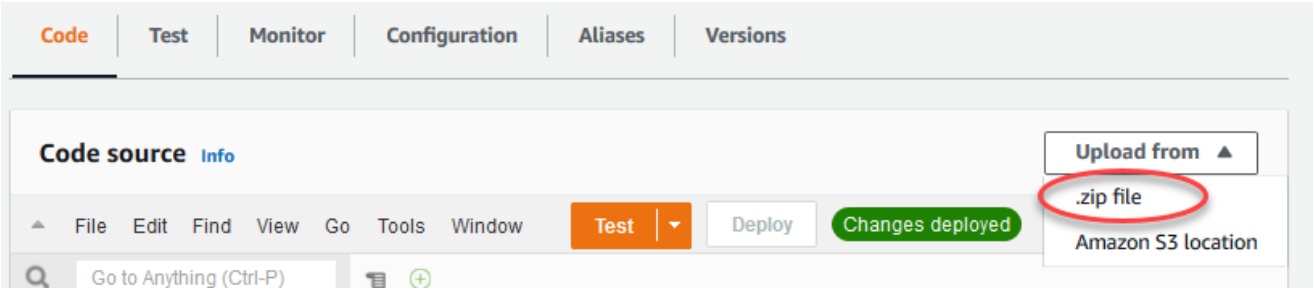
   c. Di bagian Informasi dasar tersebut, gunakan nilai-nilai berikut:

      - Untuk Nama fungsi, masukkan **`SecretTest`**.
      - Untuk Waktu pengoperasian, pilih Python 3.7.
      - Untuk Izin, pertahankan pengaturan default. Hal ini menciptakan peran eksekusi yang memberikan izin Lambda basic. Peran ini tidak digunakan oleh AWS IoT Greengrass.

   d. Di bagian bawah halaman, pilih Buat Fungsi.

2. Selanjutnya, daftarkan handler dan unggah paket deployment fungsi Lambda Anda.

   a. Pada tab Kode ini, di bawah Sumber kode, pilih Unggah dari. Dari dropdown, pilih file .zip.



   b. Pilih Mengunggah, lalu pilih paket deployment `secret_test_python.zip` Anda. Lalu, pilih Simpan.

c. Pada tab Kode fungsi, di bawah Pengaturan waktu aktif, pilih Edit, dan kemudian masukkan nilai-nilai berikut.

   - Untuk Waktu pengoperasian, pilih Python 3.7.
   - Untuk Handler, masukkan **`secret_test.function_handler`**

d. Pilih Save (Simpan).

> **Note**
>
> Tombol Tes pada konsol AWS Lambda tidak bekerja dengan fungsi ini. Pada AWS IoT Greengrass Core SDK tidak berisi modul yang diperlukan untuk menjalankan fungsi Greengrass Lambda Anda secara independen di konsol AWS Lambda tersebut. Modul-modul ini (misalnya, `greengrass_common`) dipasok ke fungsi setelah mereka di-deploy ke core Greengrass Anda.

3. Sekarang, publikasikan versi pertama fungsi Lambda Anda dan membuat alias untuk versi.

> **Note**
>
> Grup Greengrass dapat mereferensi fungsi Lambda dengan alias (direkomendasikan) atau dengan versi. Menggunakan alias membuatnya lebih mudah untuk mengelola pembaruan kode karena Anda tidak perlu mengubah tabel langganan atau definisi grup ketika kode fungsi diperbarui. Sebaliknya, Anda hanya mengarahkan alias ke versi fungsi baru.

a. Dari menu Tindakan ini, pilih Terbitkan versi baru.

b. Untuk Versi Deskripsi, masukkan **`First version`**, lalu pilih Publikasikan.

c. Pada halaman konfigurasi SecretTest: 1, dari menu Tindakan, pilih Buat alias.

d. Pada halaman Buat alias baru ini, gunakan nilai-nilai berikut:

   - Untuk Nama, masukkan **`GG_SecretTest`**.
   - Untuk Versi, pilih 1.

> **Note**
>
> AWS IoT Greengrass tidak support alias Lambda untuk versi $TERBARU ini.

e. Pilih Create (Buat).

Sekarang Anda siap untuk menambahkan fungsi Lambda ke grup Greengrass Anda dan melampirkan sumber daya rahasia.

Langkah 5: Menambahkan fungsi Lambda ke grup Greengrass

Pada langkah ini, Anda akan menambahkan fungsi Lambda ke grup Greengrass di konsol AWS IoT tersebut.

1. Pada halaman konfigurasi grup, pilih tab fungsi Lambda.
2. Di bawah bagian Fungsi Lambda Saya, pilih Tambah.
3. Untuk fungsi Lambda, pilih. SecretTest
4. Untuk versi fungsi Lambda, pilih alias ke versi yang Anda terbitkan.

Selanjutnya, konfigurasikan siklus hidup fungsi Lambda.

1. Di bagian konfigurasi fungsi Lambda, buat pembaruan berikut.

> **Note**
>
> Kami merekomendasikan Anda menjalankan fungsi Lambda Anda tanpa kontainerisasi kecuali kasus bisnis Anda memerlukannya. Hal ini membantu mengaktifkan akses ke perangkat GPU dan kamera Anda tanpa mengonfigurasi sumber daya perangkat. Jika Anda menjalankan tanpa kontainerisasi, Anda juga harus memberikan akses root ke Fungsi Lambda AWS IoT Greengrass Anda.

a. Untuk berjalan tanpa kontainerisasi:

- Untuk pengguna dan grup Sistem, pilih**`Another user ID/group ID`**. Untuk ID pengguna Sistem, masukkan**`0`**. Untuk ID grup Sistem, masukkan**`0`**.

Hal ini memungkinkan fungsi Lambda Anda untuk berjalan sebagai root. Untuk informasi lebih lanjut untuk menjalankan sebagai root, lihat the section called “Mengatur identitas akses default untuk fungsi Lambda dalam grup”.

> **Tip**
>
> Anda juga harus memperbarui file `config.json` untuk memberikan akses root ke fungsi Lambda Anda. Untuk prosedur ini, lihat the section called “Menjalankan fungsi Lambda sebagai root”.

- Untuk kontainerisasi fungsi Lambda, pilih No container.

  Untuk informasi lebih lanjut untuk berjalan tanpa konainerisasi, lihat the section called “Pertimbangan ketika memilih fungsi Lambda kontainerisasi”.
- Untuk Timeout, masukkan **`10 seconds`**.
- Untuk Pinned, pilih True.

  Untuk informasi selengkapnya, lihat the section called “Konfigurasi siklus hidup”.
- Di bawah Parameter Tambahan, untuk akses Baca ke direktori /sys, pilih Diaktifkan.

b. Untuk menjalankan dalam mode kontainer sebagai gantinya:

> **Note**
>
> Kami tidak merekomendasikan untuk dijalankan dalam mode kontainer kecuali jika kasus bisnis Anda memerlukannya.

- Untuk pengguna dan grup Sistem, pilih Gunakan grup default.
- Untuk kontainerisasi fungsi Lambda, pilih Gunakan default grup.
- Untuk Batas memori, masukkan **`1024 MB`**.
- Untuk Timeout, masukkan **`10 seconds`**.
- Untuk Pinned, pilih True.

  Untuk informasi selengkapnya, lihat the section called “Konfigurasi siklus hidup”.
- Di bawah Parameter Tambahan, untuk akses Baca ke direktori /sys, pilih Diaktifkan.

2. Pilih Tambahkan fungsi Lambda.

Selanjutnya, kaitkan sumber daya rahasia dengan fungsinya.

Langkah 6: Lampirkan sumber daya rahasia ke fungsi Lambda

Pada langkah ini, Anda mengaitkan sumber daya rahasia ke fungsi Lambda di grup Greengrass Anda. Ini mengaitkan sumber daya dengan fungsi, yang memungkinkan fungsi untuk mendapatkan nilai rahasia lokal.

1. Pada halaman konfigurasi grup, pilih tab fungsi Lambda.
2. Pilih SecretTestfungsinya.
3. Pada halaman detail fungsi, pilih Resources.
4. Gulir ke bagian Rahasia dan pilih Associate.
5. Pilih MyTestSecret, lalu pilih Associate.

Langkah 7: Menambahkan langganan ke grup Greengrass

Pada langkah ini, Anda akan menambahkan langganan yang memungkinkan AWS IoT dan fungsi Lambda untuk bertukar pesan. Satu langganan memungkinkan AWS IoT untuk memanggil fungsi, dan memungkinkan fungsi untuk mengirim data output ke AWS IoT.

1. Pada halaman konfigurasi grup, pilih tab Langganan, lalu pilih Tambah Langganan.
2. Buat langganan yang mengizinkan AWS IoT untuk mempublikasikan pesan ke fungsi.

   Pada halaman konfigurasi grup, pilih tab Langganan, lalu pilih Tambahkan langganan.
3. Pada halaman Buat langganan, konfigurasikan sumber dan target, sebagai berikut:

   a. Pada tipe Sumber, pilih fungsi Lambda, lalu pilih IoT Cloud.

   b. Di Jenis target, pilih Layanan, lalu pilih SecretTest.

   c. Di filter Topik, masukkan**`secrets/input`**, lalu pilih Buat langganan.
4. Tambahkan langganan kedua. Pilih tab Langganan, pilih Tambahkan langganan, dan konfigurasikan sumber dan target, sebagai berikut:

   a. Di tipe Sumber, pilih Layanan, lalu pilih SecretTest.

   b. Pada tipe Target, pilih fungsi Lambda, lalu pilih IoT Cloud.

   c. Di filter Topik, masukkan**`secrets/output`**, lalu pilih Buat langganan.

Langkah 8: Men-deploy grup Greengrass

Men-deploy grup ke perangkat core. Selama deployment, AWS IoT Greengrass mengambil nilai rahasia dari Secrets Manager dan membuat salinan lokal yang dienkripsi pada core.

1. Pastikan bahwa AWS IoT Greengrass core sedang berjalan. Jalankan perintah berikut di terminal Raspberry Pi Anda, sesuai kebutuhan.

   a. Untuk memeriksa apakah daemon sedang berjalan:

   ```
   ps aux | grep -E 'greengrass.*daemon'
   ```

   Jika outputnya berisi entri `root` untuk `/greengrass/ggc/packages/`*`ggc-version`*`/bin/daemon`, maka daemon sedang berjalan.

   > **Note**
   >
   > Versi di jalur tergantung pada versi perangkat lunak AWS IoT Greengrass core yang diinstal pada perangkat core Anda.

   b. Untuk memulai daemon:

   ```
   cd /greengrass/ggc/core/
   sudo ./greengrassd start
   ```

2. Pada halaman konfigurasi grup, pilih Deploy.

3. a. Di tab Fungsi Lambda, di bawah bagian Fungsi Lambda Sistem, pilih detektor IP dan pilih Edit.

   b. Dalam kotak dialog Edit pengaturan detektor IP, pilih Secara otomatis mendeteksi dan mengganti titik akhir broker MQTT.

   c. Pilih Save (Simpan).

   Hal ini mengaktifkan perangkat untuk secara otomatis memperoleh informasi konektivitas untuk core, seperti alamat IP, DNS, dan nomor port. Deteksi otomatis direkomendasikan, namun AWS IoT Greengrass juga support titik akhir yang ditentukan secara manual. Anda hanya diminta untuk metode penemuan pertama kalinya bahwa grup di-deploy.

> **Note**
>
> Jika diminta, berikan izin untuk membuat Peran layanan Greengrass dan kaitkan dengan Akun AWS Anda pada Wilayah AWS. Peran ini memungkinkan AWS IoT Greengrass untuk mengakses sumber daya Anda di layanan AWS ini.

Halaman Deployment menampilkan timestamp deployment, ID versi, dan status. Setelah selesai, status yang ditampilkan untuk penerapan harus Selesai.

Untuk bantuan penyelesaian masalah, lihat Memecahkan masalah.

Tes fungsi Lambda

1. Di halaman beranda konsol AWS IoT tersebut, pada panel sebelah kiri, pilih Tes.
2. Untuk Berlangganan topik, gunakan nilai-nilai berikut, dan kemudian pilih Langganan.

Properti	Nilai
Topik langganan	rahasia/output
Tampilan muatan MQTT	Tampilkan muatan sebagai string

3. Untuk Publikasikan ke topik, gunakan nilai-nilai berikut, dan kemudian pilih Publikasikan untuk memanggil fungsi.

Properti	Nilai
Topik	rahasia/input
Message	Pertahankan pesan default. Publikasi pesan memanggil fungsi Lambda, tetapi fungsi dalam tutorial ini tidak memproses tubuh pesan.

Jika berhasil, fungsi menunjukkan pesan "Sukses".

Lihat juga

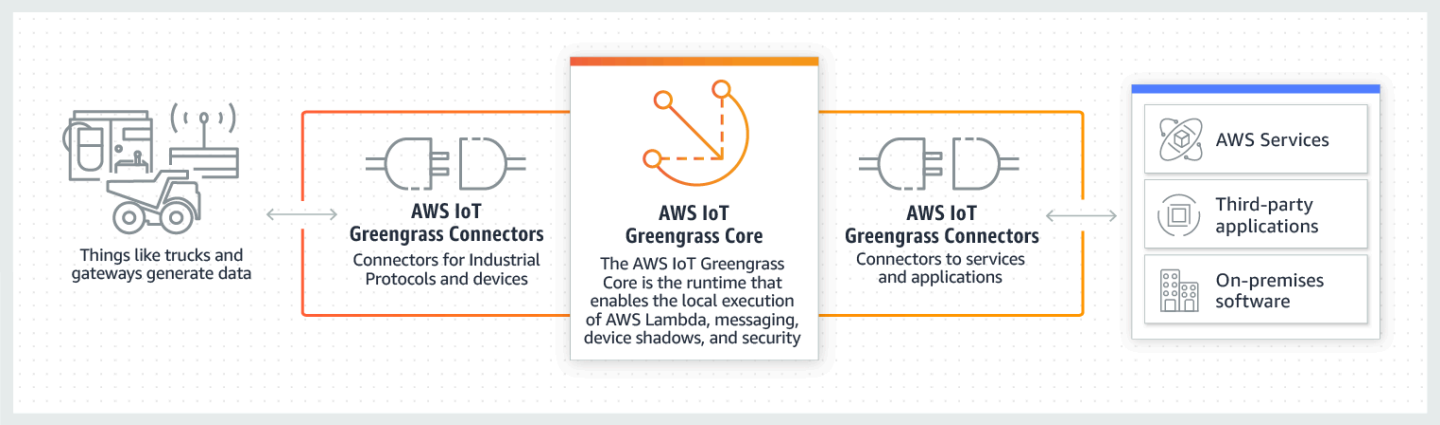
- Men-deploy rahasia ke core

Integrasikan dengan layanan dan protokol menggunakan konektor Greengrass

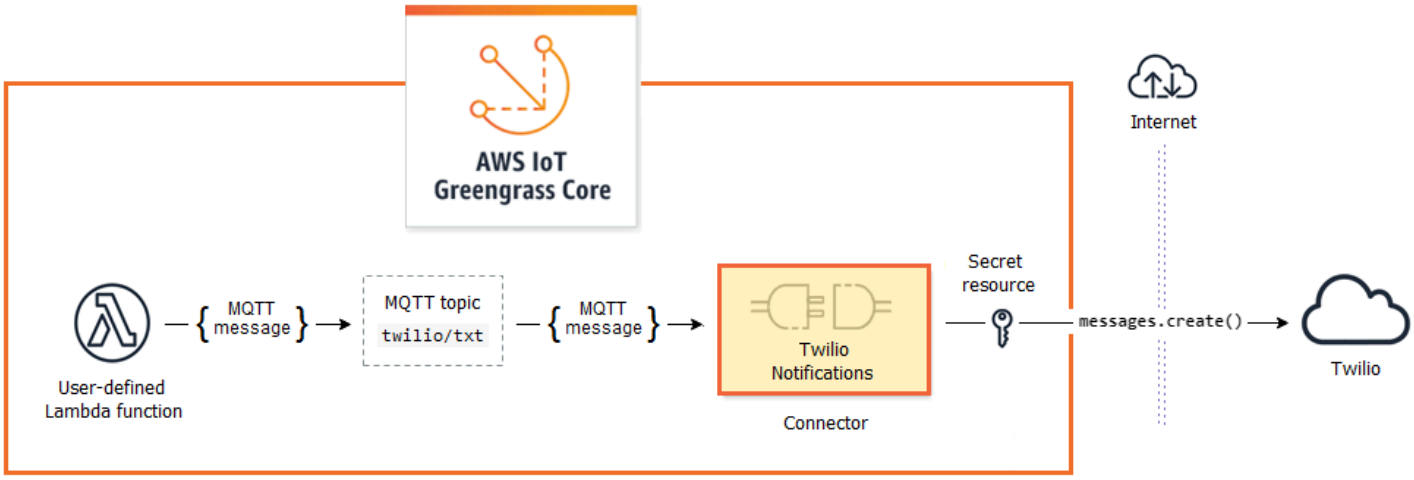
Fitur ini tersedia untuk AWS IoT Greengrass Core v1.7 dan yang lebih baru.

Konektor di AWS IoT Greengrass adalah modul prebuilt yang membuatnya lebih efisien untuk berinteraksi dengan infrastruktur lokal, protokol perangkat, AWS, dan layanan cloud lainnya. Dengan menggunakan konektor, Anda dapat menghabiskan lebih sedikit waktu untuk mempelajari protokol dan API baru dan lebih banyak waktu fokus pada logika yang berarti bagi bisnis Anda.

Diagram berikut menunjukkan di mana konektor dapat masuk ke dalam AWS IoT Greengrass lanskap.



Banyak konektor menggunakan pesan MQTT untuk berkomunikasi dengan perangkat klien dan fungsi Lambda Greengrass di grup, atau denganAWS IoTdan layanan bayangan lokal. Pada contoh berikut, konektor Notifikasi Twilio menerima pesan MQTT dari fungsi Lambda yang ditetapkan pengguna, menggunakan referensi lokal rahasia dari AWS Secrets Manager, dan memanggil Twilio API.



Untuk tutorial yang membuat solusi ini, lihat the section called “Memulai dengan konektor (konsol)” dan the section called “Memulai dengan konektor (CLI)”.

Konektor Greengrass dapat membantu Anda memperluas kemampuan perangkat atau membuat perangkat tujuan tunggal. Dengan menggunakan konektor, Anda dapat:

- Menerapkan logika bisnis dapat digunakan kembali.
- Berinteraksi dengan layanan cloud dan lokal, termasuk AWS dan layanan pihak ketiga.
- Menelan dan memproses data peranti.
- Aktifkan device-to-device panggilan menggunakan langganan topik MQTT dan fungsi Lambda yang ditetapkan pengguna.

AWS menyediakan satu set konektor Greengrass yang menyederhanakan interaksi dengan layanan umum dan sumber data. Modul prebuilt ini memungkinkan skenario untuk pencatatan dan diagnostik, penambahan, pemrosesan data industri, dan alarm dan pesan. Untuk informasi selengkapnya, lihat the section called “AWS-disediakan konektor Greengrass”.

Persyaratan

Untuk menggunakan konektor, ingatlah hal-hal ini:

- Setiap konektor yang Anda gunakan memiliki persyaratan yang harus Anda penuhi. Persyaratan ini mungkin termasuk minimum AWS IoT Greengrass Versi perangkat lunak Core, prasyarat

perangkat, izin yang diperlukan, dan batas. Untuk informasi selengkapnya, lihat the section called "AWS-disediakan konektor Greengrass".

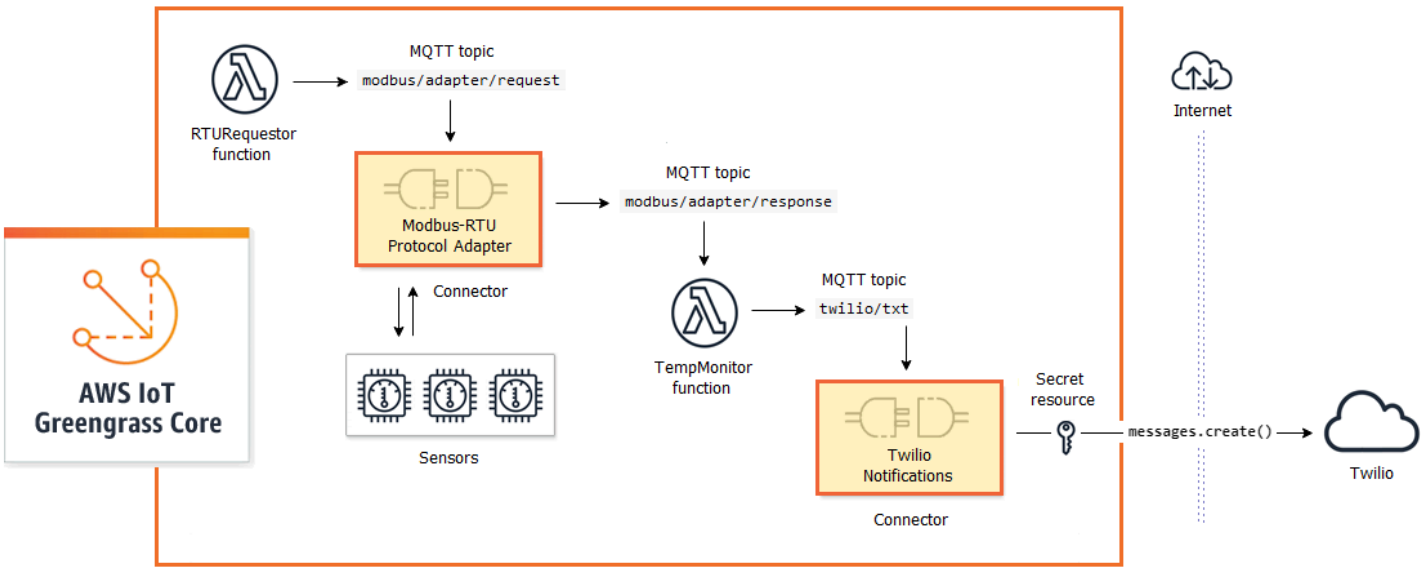
- Sebuah grup Greengrass dapat berisi hanya satu contoh dikonfigurasi dari konektor yang diberikan. Namun, Anda dapat menggunakan contoh dalam beberapa langganan. Untuk informasi selengkapnya, lihat the section called "Parameter konfigurasi".
- Ketika kontainerisasi default untuk grup Greengrass diatur ke Tanpa kontainer, konektor di grup harus berjalan tanpa kontainerisasi. Untuk menemukan konektor yang mendukung mode Tanpa kontainer ini, lihat the section called "AWS-disediakan konektor Greengrass".

Menggunakan konektor Greengrass

Konektor adalah jenis komponen grup. Seperti komponen grup lainnya, seperti perangkat klien dan fungsi Lambda yang ditetapkan pengguna, Anda menambahkan konektor ke grup, mengonfigurasi pengaturan mereka, dan menerapkannya ke grup, mengonfigurasi pengaturan mereka, dan menerapkannya keAWS IoT Greengrassinti. Konektor berjalan di lingkungan core.

Anda dapat menggunakan beberapa konektor sebagai aplikasi mandiri sederhana. Sebagai contoh, konektor Device Defender membaca metrik sistem dari perangkat core dan mengirimkannya AWS IoT Device Defender untuk analisis.

Anda dapat menambahkan konektor lain sebagai blok bangunan dalam solusi yang lebih besar. Contoh berikut solusi menggunakan konektor Adaptor ModBus-RTU untuk memproses pesan dari sensor dan konektor Notifikasi Twilio untuk memulai pesan Twilio.



Solusi sering mencakup fungsi Lambda yang ditetapkan pengguna yang duduk di samping konektor dan memproses data yang dikirim atau diterima konektor. Dalam contoh ini, TempMonitorfungsi menerima data dari ModBus-RTU Protocol Adapter, menjalankan beberapa logika bisnis, dan kemudian mengirimkan data ke Notifikasi Twilio.

Untuk membuat dan menerapkan solusi, Anda mengikuti proses umum ini:

1. Memetakan aliran data tingkat tinggi. Mengidentifikasi sumber data, saluran data, layanan, protokol, dan sumber daya yang Anda butuhkan untuk bekerja dengan. Dalam solusi contoh, ini termasuk data melalui protokol Modbus RTU, port serial Modbus fisik, dan Twilio.
2. Identifikasi konektor untuk disertakan dalam solusi, dan tambahkan ke grup Anda. Solusi contoh menggunakan Modbus-RTU Protocol Adapter dan Notifikasi Twilio. Untuk membantu Anda menemukan konektor yang berlaku untuk skenario Anda, dan untuk mempelajari tentang persyaratan masing-masing, lihat the section called “AWS-disediakan konektor Greengrass”.
3. Mengidentifikasi apakah fungsi Lambda pengguna yang ditentukan, perangkat, atau sumber daya yang diperlukan, dan kemudian membuat dan menambahkannya ke grup. Ini mungkin termasuk fungsi yang berisi logika bisnis atau data proses ke dalam format yang diperlukan oleh entitas lain dalam solusi. Contoh solusi menggunakan fungsi untuk mengirim permintaan Modbus RTU dan memulai notifikasi Twilio. Ini juga mencakup sumber daya perangkat lokal untuk port serial Modbus RTU dan sumber daya rahasia untuk token otentikasi Twilio.

   > **Note**
   >
   > Password referensi sumber daya rahasia, token, dan rahasia lainnya dari AWS Secrets Manager. Rahasia dapat digunakan oleh konektor dan fungsi Lambda untuk mengotentikasi dengan layanan dan aplikasi. Secara default, AWS IoT Greengrass dapat mengakses rahasia dengan nama yang dimulai dengan "greengrass-". Untuk informasi selengkapnya, lihat Men-deploy rahasia ke core.

4. Buat langganan yang mengizinkan entitas dalam solusi untuk bertukar pesan MQTT. Jika konektor digunakan di langganan, konektor dan sumber pesan atau target harus menggunakan sintaks topik yang telah ditetapkan yang didukung oleh konektor. Untuk informasi selengkapnya, lihat the section called “Input dan output”.
5. Terapkan grup ke Greengrass core.

Untuk informasi tentang membuat dan menerapkan konektor, lihat tutorial berikut:

- the section called “Memulai dengan konektor (konsol)”
- the section called “Memulai dengan konektor (CLI)”

Parameter konfigurasi

Banyak konektor menyediakan parameter yang memungkinkan Anda menyesuaikan perilaku atau output. Parameter ini digunakan selama inisialisasi, di runtime, atau di lain waktu dalam siklus hidup konektor.

Jenis dan penggunaan parameter bervariasi menurut konektor. Sebagai contoh, penyambung SNS mempunyai parameter yang mengonfigurasi topik default SNS, dan Device Defender mempunyai parameter yang mengonfigurasi tingkat sampling data.

Sebuah versi grup dapat berisi beberapa konektor, tetapi hanya satu contoh dari konektor yang diberikan pada suatu waktu. Ini berarti bahwa setiap konektor dalam grup hanya dapat memiliki satu konfigurasi aktif. Namun, konektor instance dapat digunakan dalam beberapa langganan di grup. Sebagai contoh, Anda dapat membuat langganan yang mengizinkan banyak perangkat mengirim data ke konektor Kinesis Firehose.

Parameter yang digunakan untuk mengakses sumber daya grup

Greengrass konektor menggunakan sumber daya grup untuk mengakses sistem file, port, periferal, dan sumber daya lokal lainnya pada perangkat core. Jika konektor memerlukan akses ke sumber daya kelompok, maka konektor menyediakan parameter konfigurasi terkait.

Sumber daya grup meliputi:

- Sumber daya lokal. Direktori, file, port, pin, dan periferal yang hadir pada perangkat Greengrass core.
- Sumber daya machine learning. Model machine learning yang dilatih di cloud dan diterapkan ke core untuk inferensi lokal.
- Sumber daya rahasia. Lokal. salinan enkripsi password, kunci, token, atau teks arbitrer dari AWS Secrets Manager. Konektor dapat dengan aman mengakses rahasia lokal ini dan menggunakannya untuk mengotentikasi ke layanan atau infrastruktur lokal.

Sebagai contoh, parameter untuk Device Defender memungkinkan akses ke metrik sistem di direktori `/proc` host, dan parameter untuk Notifikasi Twilio memungkinkan akses ke token otentikasi Twilio yang disimpan secara lokal.

Memperbarui parameter konektor

Parameter dikonfigurasi saat konektor ditambahkan ke grup Greengrass. Anda dapat mengubah nilai parameter setelah konektor ditambahkan.

- Di konsol: Dari halaman konfigurasi grup, bukaKonektor, dan dari menu kontekstual konektor, pilihDiedit.

  > **Note**
  >
  > Jika konektor menggunakan sumber daya rahasia yang kemudian diubah untuk referensi rahasia yang berbeda, Anda harus mengedit parameter konektor dan mengonfirmasi perubahan.

- Di API: Buat versi lain dari konektor yang mendefinisikan konfigurasi baru.

  API AWS IoT Greengrass menggunakan versi untuk mengelola grup. Versi tidak dapat diubah, sehingga untuk menambah atau mengubah komponen grup—misalnya, perangkat klien grup, dan sumber daya—Anda harus membuat versi komponen baru atau yang diperbarui. Kemudian, Anda buat dan terapkan versi grup yang berisi versi target masing-masing komponen.

Setelah Anda membuat perubahan konfigurasi konektor, Anda harus menerapkan grup untuk memperbanyak perubahan ke core.

Input dan output

Banyak konektor Greengrass dapat berkomunikasi dengan entitas lain dengan mengirim dan menerima pesan MQTT. Komunikasi MQTT dikendalikan oleh langganan yang memungkinkan konektor untuk bertukar data dengan fungsi Lambda, perangkat klien, dan konektor lainnya dalam grup Greengrass, atau denganAWS IoTdan layanan bayangan lokal. Untuk mengizinkan komunikasi ini, Anda harus membuat langganan dalam grup yang menjadi milik konektornya. Untuk informasi selengkapnya, lihat the section called “Langganan yang dikelola di dalam alur kerja pesan MQTT”.

Konektor dapat penerbit pesan, pelanggan pesan, atau keduanya. Setiap konektor mendefinisikan topik MQTT yang menerbitkan atau berlangganan. Topik standar ini harus digunakan dalam langganan di mana konektor adalah sumber pesan atau target pesan. Untuk tutorial yang menyertakan langkah-langkah untuk mengonfigurasi langganan untuk konektor, lihat the section called “Memulai dengan konektor (konsol)” dan the section called “Memulai dengan konektor (CLI)”.

> **Note**
>
> Banyak konektor juga memiliki mode komunikasi built-in untuk berinteraksi dengan layanan cloud atau lokal. Ini bervariasi oleh konektor dan mungkin mengharuskan Anda mengonfigurasi parameter atau menambahkan izin untuk peran grup. Untuk informasi tentang persyaratan konektor, lihat the section called “AWS-disediakan konektor Greengrass”.

Topik input

Kebanyakan konektor menerima data input mengenai topik MQTT. Beberapa konektor berlangganan beberapa topik untuk input data. Sebagai contoh, konektor Serial Stream mendukung dua topik:

- `serial/+/read/#`
- `serial/+/write/#`

Untuk konektor ini, permintaan baca dan tulis dikirim ke topik yang sesuai. Ketika Anda membuat langganan, pastikan untuk menggunakan topik yang sesuai dengan implementasi Anda.

Karakter + dan # di contoh sebelumnya adalah wildcard. Wildcard ini memungkinkan pelanggan untuk menerima pesan pada beberapa topik dan penerbit untuk menyesuaikan topik yang mereka terbitkan.

- Wildcard + dapat muncul di mana saja dalam hierarki topik. Hal ini dapat diganti dengan satu item hirarki.

  Sebagai contoh, untuk topik `sensor/+/input`, pesan dapat diterbitkan ke topik `sensor/id-123/input` tetapi tidak untuk `sensor/group-a/id-123/input`.

- Wildcard # dapat muncul hanya pada akhir hierarki topik. Hal ini dapat diganti dengan nol atau lebih item hirarki.

Sebagai contoh, untuk topik `sensor/#`, pesan dapat diterbitkan ke `sensor/`, `sensor/id-123`, dan `sensor/group-a/id-123`, tapi tidak untuk `sensor`.

Karakter wildcard hanya berlaku saat berlangganan topik. Pesan tidak dapat diterbitkan ke topik yang berisi wildcard. Periksa dokumentasi untuk konektor untuk informasi lebih lanjut tentang persyaratan topik input atau output. Untuk informasi selengkapnya, lihat the section called “AWS-disediakan konektor Greengrass”.

Dukungan kontainerisasi

Secara default, sebagian besar konektor berjalan pada Greengrass core dalam lingkungan runtime terisolasi yang dikelola oleh AWS IoT Greengrass. Lingkungan runtime ini, disebut kontainer, menyediakan isolasi antara konektor dan sistem host, yang menawarkan lebih banyak keamanan untuk host dan konektor.

Namun, kontainerisasi Greengrass ini tidak didukung di beberapa lingkungan, seperti ketika Anda menjalankan AWS IoT Greengrass dalam kontainer Docker atau kernel Linux yang lebih tua tanpa cgroups. Dalam lingkungan ini, konektor harus berjalan dalam di mode Tanpa kontainer. Untuk menemukan konektor yang mendukung mode Tanpa kontainer ini, lihat the section called “AWS-disediakan konektor Greengrass”. Beberapa konektor berjalan dalam mode ini secara native, dan beberapa konektor memungkinkan Anda mengatur mode isolasi.

Anda juga dapat mengatur mode isolasi ke Tanpa kontainer di lingkungan yang mendukung kontainerisasi Greengrass, tetapi kami sarankan menggunakan mode kontainer Greengrass bila memungkinkan.

> **Note**
>
> Kontainerisasi default untuk grup Greengrass tidak berlaku untuk konektor.

Versi upgrade konektor

Penyedia konektor mungkin merilis konektor versi baru yang menambahkan fitur, memperbaiki masalah, atau meningkatkan kinerja. Untuk informasi tentang versi yang tersedia dan perubahan terkait, lihat dokumentasi untuk setiap konektor.

Di konsol AWS IoT tersebut, Anda dapat memeriksa versi baru untuk konektor dalam grup Greengrass Anda.

1. DiAWS IoTpanel navigasi konsol, di bawahKelola, PerluasPerangkat Greengrass, lalu pilihGrup (V1).
2. Di bawah grup Greengrass, pilih grup Anda.
3. Pilih Konektor untuk menampilkan konektor dalam grup.

   Jika konektor memiliki versi baru, sebuah tombol Tersedia muncul di kolom Upgrade ini.
4. Untuk upgrade versi konektor:

   a. Di halaman Konektor ini, pada kolom Upgrade ini, pilih Tersedia. Halaman Upgrade konektor membuka dan menampilkan pengaturan parameter saat ini, jika berlaku.

      Pilih versi konektor baru, tentukan parameter sesuai kebutuhan, lalu pilih Upgrade.

   b. Di halaman Langganan, tambahkan langganan baru di grup untuk mengganti yang menggunakan konektor sebagai sumber atau target. Kemudian, hapus langganan lama.

      Konektor referensi langganan berdasarkan versi, sehingga merekan menjadi tidak valid jika Anda mengubah versi konektor di grup.

   c. Dari menu Tindakan ini, pilih Deploy untuk menerapkan perubahan Anda ke core.

Untuk upgrade konektor dari AWS IoT Greengrass API, buat dan terapkan versi grup yang mencakup konektor di-upgrade dan langganan. Gunakan proses yang sama seperti ketika Anda menambahkan konektor ke grup. Untuk langkah-langkah mendetail yang menunjukkan bagaimana menggunakan AWS CLI untuk mengonfigurasi dan menerapkan contoh konektor Notifikasi Twilio, lihat the section called “Memulai dengan konektor (CLI)”.

Pencatatan untuk konektor

Konektor Greengrass berisi fungsi Lambda yang menulis peristiwa dan kesalahan untuk Greengrass logs. Tergantung pada pengaturan grup Anda, log ditulis ke CloudWatch Log, sistem file lokal, atau keduanya. Log dari konektor termasuk ARN dari fungsi yang sesuai. Contoh ARN berikut adalah dari konektor Kinesis Firehose:

```
arn:aws:lambda:aws-region:account-id:function:KinesisFirehoseClient:1
```

Konfigurasi pencatatan default menulis log info-level ke file yang menggunakan struktur direktori berikut:

```
greengrass-root/ggc/var/log/user/region/aws/function-name.log
```

Untuk informasi selengkapnya tentang pencatatan Greengrass, lihat the section called “Pemantauan dengan AWS IoT Greengrass log”.

AWS-disediakan konektor Greengrass

AWS menyediakan konektor berikut yang mendukung AWS IoT Greengrass skenario umum. Untuk informasi lebih lanjut tentang cara kerja konektor, lihat dokumentasi berikut:

- Integrasikan dengan layanan dan protokol menggunakan konektor
- Memulai dengan konektor (konsol) atau Memulai dengan konektor (CLI)

Konektor	Deskripsi	Lambda waktu aktif yang didukung	Mendukung mode Tanpa kontainer
CloudWatch Metrik	Menerbitkan metrik khusus ke Amazon. CloudWatch	• Python 3.8 * • Python 3.7 • Python 2.7	Ya
Pembela Perangkat	Mengirim metrik sistem ke AWS IoT Device Defender.	• Python 3.8 * • Python 3.7 • Python 2.7	Tidak
Penerapan Aplikasi Docker	Jalankan file Docker Compose untuk memulai aplikasi Docker pada perangkat core.	• Python 3.8 • Python 3.7	Ya
IoT Analytics	Mengirim data dari perangkat dan sensor ke AWS IoT Analytics.	• Python 3.8 *	Ya

Konektor	Deskripsi	Lambda waktu aktif yang didukung	Mendukung mode Tanpa kontainer
		• Python 3.7 • Python 2.7	
Adaptor Protokol IP IoT Ethernet	Mengumpulkan data dari perangkat Ethernet/IP.	• Java 8	Ya
IoT SiteWise	Mengirimkan data dari perangkat dan sensor ke properti aset dalam AWS IoT SiteWise.	• Java 8	Ya
Kinesis Firehose	Mengirim data ke aliran pengiriman Amazon Data Firehose.	• Python 3.8 * • Python 3.7 • Python 2.7	Ya
Umpan Balik	Menerbitkan machine learning model input ke cloud dan output ke topik MQTT.	• Python 3.8 • Python 3.7	Tidak
Klasifikasi Gambar ML	Menjalankan layanan inferensi klasifikasi gambar lokal. Konektor ini menyediakan versi untuk beberapa platform.	• Python 3.8 * • Python 3.7 • Python 2.7	Tidak
Deteksi Objek ML	Menjalankan sebuah layanan inferensi deteksi objek lokal. Konektor ini menyediakan versi untuk beberapa platform.	• Python 3.8 • Python 3.7	Tidak
Adaptor Protokol Modbus-RTU	Mengirim permintaan ke perangkat Modbus RTU.	• Python 3.8 * • Python 3.7 • Python 2.7	Tidak

Konektor	Deskripsi	Lambda waktu aktif yang didukung	Mendukung mode Tanpa kontainer
Adaptor Protokol Modbus-TCP	Mengumpulkan data dari perangkat ModBustCP.	• Java 8	Ya
Raspberry Pi GPIO	Kontrol GPIO pin pada Raspberry Pi core.	• Python 3.8 * • Python 3.7 • Python 2.7	Tidak
Aliran Serial	Membaca dan menulis ke port serial pada perangkat core.	• Python 3.8 * • Python 3.7 • Python 2.7	Tidak
ServiceNow MetricBase Integrasi	Menerbitkan metrik deret waktu ke. ServiceNow MetricBase	• Python 3.8 * • Python 3.7 • Python 2.7	Ya
SNS	Mengirim pesan ke topik Amazon SNS.	• Python 3.8 * • Python 3.7 • Python 2.7	Ya
Integrasi Splunk	Menerbitkan data untuk Splunk HEC.	• Python 3.8 * • Python 3.7 • Python 2.7	Ya

Konektor	Deskripsi	Lambda waktu aktif yang didukung	Mendukung mode Tanpa kontainer
Pemberitahuan Twilio	Memulai teks Twilio atau pesan suara.	• Python 3.8 * • Python 3.7 • Python 2.7	Ya

* Untuk menggunakan waktu aktifs Python 3.8, Anda harus membuat sebuah link simbolik dari folder instalasi default Python 3.7 ke binari Python 3.8 yang terinstal. Untuk informasi lebih lanjut, lihat persyaratan khusus konektor.

> **Note**
>
> Kami merekomendasikan bahwa Anda memperbarui versi konektor dari Python 2.7 ke Python 3.7. Dukungan berkelanjutan untuk konektor Python 2.7 tergantung pada AWS Lambda dukungan runtime. Untuk informasi selengkapnya, lihat Kebijakan dukungan waktu aktif di Panduan Developer AWS Lambda .

CloudWatch Konektor metrik

Konektor CloudWatch Metrik menerbitkan metrik khusus dari perangkat Greengrass ke Amazon. CloudWatch Konektor menyediakan infrastruktur terpusat untuk menerbitkan CloudWatch metrik, yang dapat Anda gunakan untuk memantau dan menganalisis lingkungan inti Greengrass, dan bertindak berdasarkan acara lokal. Untuk informasi selengkapnya, lihat Menggunakan CloudWatch metrik Amazon di Panduan CloudWatch Pengguna Amazon.

Konektor ini menerima data metrik sebagai pesan MQTT. Konektor mengumpulkan metrik yang berada di namespace yang sama dan menerbitkannya secara berkala. CloudWatch

Konektor ini memiliki versi berikut.

Versi	ARN
5	`arn:aws:greengrass:region::/connectors/CloudWatchMetrics/versions/5`
4	`arn:aws:greengrass:region::/connectors/CloudWatchMetrics/versions/4`
3	`arn:aws:greengrass:region::/connectors/CloudWatchMetrics/versions/3`
2	`arn:aws:greengrass:region::/connectors/CloudWatchMetrics/versions/2`
1	`arn:aws:greengrass:region::/connectors/CloudWatchMetrics/versions/1`

Untuk informasi tentang perubahan versi, lihat Changelog.

Persyaratan

Konektor ini memiliki persyaratan sebagai berikut:

Version 3 - 5

- AWS IoT Greengrass perangkat lunak Core v1.9.3 atau yang lebih baru.
- Python versi 3.7 atau 3.8 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.

> **Note**
>
> Untuk menggunakan Python 3.8, jalankan perintah berikut untuk membuat link simbolik dari folder instalasi default Python 3.7 ke binari Python 3.8 yang diinstal.

```
sudo ln -s path-to-python-3.8/python3.8 /usr/bin/python3.7
```

Ini mengonfigurasi perangkat Anda untuk memenuhi persyaratan Python untuk AWS IoT Greengrass.

- Pada Peran grup Greengrass dikonfigurasi untuk mengizinkan `cloudwatch:PutMetricData` tindakan, seperti yang ditunjukkan dalam contoh berikut AWS Identity and Access Management kebijakan (IAM).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1528133056761",
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

Untuk persyaratan peran grup, Anda harus mengonfigurasi peran untuk memberikan izin yang diperlukan dan memastikan peran telah ditambahkan ke grup. Untuk informasi lebih lanjut, lihat the section called “Mengelola peran grup (konsol)” atau the section called “Kelola peran grup (CLI)”.

Untuk informasi selengkapnya tentang CloudWatch izin, lihat referensi CloudWatch izin Amazon di Panduan Pengguna IAM.

Versions 1 - 2

- AWS IoT Greengrass perangkat lunak Core v1.7 atau yang lebih baru.
- Python versi 2.7 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.
- Pada Peran grup Greengrass dikonfigurasi untuk mengizinkan `cloudwatch:PutMetricData` tindakan, seperti yang ditunjukkan dalam contoh berikut AWS Identity and Access Management kebijakan (IAM).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1528133056761",
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

Untuk persyaratan peran grup, Anda harus mengonfigurasi peran untuk memberikan izin yang diperlukan dan memastikan peran telah ditambahkan ke grup. Untuk informasi lebih lanjut, lihat the section called “Mengelola peran grup (konsol)” atau the section called “Kelola peran grup (CLI)”.

Untuk informasi selengkapnya tentang CloudWatch izin, lihat referensi CloudWatch izin Amazon di Panduan Pengguna IAM.

Parameter Konektor

Konektor ini menyediakan parameter berikut:

Versions 4 - 5

`PublishInterval`

Jumlah maksimum detik untuk menunggu sebelum menerbitkan metrik batched untuk namespace tertentu. Nilai maksimumnya adalah 900. Untuk mengonfigurasi konektor untuk menerbitkan metrik seperti yang diterima (tanpa batching), tentukan 0.

Konektor mempublikasikan CloudWatch setelah menerima 20 metrik di namespace yang sama atau setelah interval yang ditentukan.

> **Note**
>
> Konektor tidak menjamin urutan acara terbitan.

Nama tampilan pada konsol AWS IoT tersebut: Publikasikan Interval

Wajib: `true`

Jenis: `string`

Nilai yang valid: `0 - 900`

Pola yang valid: `[0-9]|[1-9]\d|[1-9]\d\d|900`

`PublishRegion`

Wilayah AWSUntuk memposting CloudWatch metrik ke. Nilai ini menimpa Wilayah metrik Greengrass default. Ini diperlukan hanya saat memposting metrik lintas-Wilayah.

Nama tampilan pada konsol AWS IoT tersebut: Publikasikan wilayah

Wajib: `false`

Jenis: `string`

Pola yang valid: `^$|([a-z]{2}-[a-z]+-\d{1})`

`MemorySize`

Memori (dalam KB) untuk mengalokasikan ke konektor.

Nama tampilan pada konsol AWS IoT tersebut: Ukuran memori

Wajib: `true`

Jenis: `string`

Pola yang valid: `^[0-9]+$`

`MaxMetricsToRetain`

Jumlah maksimum metrik dalam semua namespaces untuk dipertahankan dalam memori sebelum diganti dengan metrik baru. Nilai minimumnya adalah 2000.

Batas ini berlaku bila tidak ada koneksi ke internet dan konektor mulai buffer metrik untuk diterbitkan nanti. Ketika buffer penuh, metrik terlama digantikan oleh metrik baru. Metrik dalam namespace tertentu diganti hanya dengan metrik dalam namespace yang sama.

> **Note**
>
> Metrik tidak dipertahankan jika proses host untuk konektor terganggu. Sebagai contoh, gangguan ini dapat terjadi selama deployment grup atau ketika perangkat dimulai ulang.

Nama tampilan pada konsol AWS IoT tersebut: Metrik maksimum untuk dipertahankan

Wajib: `true`

Jenis: `string`

Pola yang valid: `^([2-9]\d{3}|[1-9]\d{4,})$`

`IsolationMode`

Mode kontainerisasi untuk konektor ini. Default-nya adalah `GreengrassContainer`, yang berarti bahwa konektor berjalan dalam lingkungan waktu aktif terisolasi dalam kontainer AWS IoT Greengrass ini.

> **Note**
>
> Pengaturan kontainerisasi default untuk grup tidak berlaku untuk konektor.

Nama tampilan pada konsol AWS IoT tersebut: Mode isolasi kontainer

Wajib: `false`

Jenis: `string`

Nilai yang valid: `GreengrassContainer` or `NoContainer`

Pola yang valid: `^NoContainer$|^GreengrassContainer$`

Versions 1 - 3

`PublishInterval`

Jumlah maksimum detik untuk menunggu sebelum menerbitkan metrik batched untuk namespace tertentu. Nilai maksimumnya adalah 900. Untuk mengonfigurasi konektor untuk menerbitkan metrik seperti yang diterima (tanpa batching), tentukan 0.

Konektor mempublikasikan CloudWatch setelah menerima 20 metrik di namespace yang sama atau setelah interval yang ditentukan.

> **Note**
>
> Konektor tidak menjamin urutan acara terbitan.

Nama tampilan pada konsol AWS IoT tersebut: Publikasikan Interval

Wajib: `true`

Jenis: `string`

Nilai yang valid: `0 - 900`

Pola yang valid: `[0-9]|[1-9]\d|[1-9]\d\d|900`

`PublishRegion`

Wilayah AWSUntuk memposting CloudWatch metrik ke. Nilai ini menimpa Wilayah metrik Greengrass default. Ini diperlukan hanya saat memposting metrik lintas-Wilayah.

Nama tampilan pada konsol AWS IoT tersebut: Publikasikan wilayah

Wajib: `false`

Jenis: `string`

Pola yang valid: `^$|([a-z]{2}-[a-z]+-\d{1})`

`MemorySize`

Memori (dalam KB) untuk mengalokasikan ke konektor.

Nama tampilan pada konsol AWS IoT tersebut: Ukuran memori

Wajib: `true`

Jenis: `string`

Pola yang valid: `^[0-9]+$`

`MaxMetricsToRetain`

Jumlah maksimum metrik dalam semua namespaces untuk dipertahankan dalam memori sebelum diganti dengan metrik baru. Nilai minimumnya adalah 2000.

Batas ini berlaku bila tidak ada koneksi ke internet dan konektor mulai buffer metrik untuk diterbitkan nanti. Ketika buffer penuh, metrik terlama digantikan oleh metrik baru. Metrik dalam namespace tertentu diganti hanya dengan metrik dalam namespace yang sama.

> **Note**
>
> Metrik tidak dipertahankan jika proses host untuk konektor terganggu. Sebagai contoh, gangguan ini dapat terjadi selama deployment grup atau ketika perangkat dimulai ulang.

Nama tampilan pada konsol AWS IoT tersebut: Metrik maksimum untuk dipertahankan

Wajib: `true`

Jenis: `string`

Pola yang valid: `^([2-9]\d{3}|[1-9]\d{4,})$`

Buat Contoh Konektor (AWS CLI)

Perintah CLI berikut membuat `ConnectorDefinition` dengan versi awal yang berisi konektor CloudWatch Metrik.

```
aws greengrass create-connector-definition --name MyGreengrassConnectors --initial-
version '{
    "Connectors": [
        {
```

```
            "Id": "MyCloudWatchMetricsConnector",
            "ConnectorArn": "arn:aws:greengrass:region::/connectors/CloudWatchMetrics/
versions/4",
            "Parameters": {
                "PublishInterval" : "600",
                "PublishRegion" : "us-west-2",
                "MemorySize" : "16",
                "MaxMetricsToRetain" : "2500",
                "IsolationMode" : "GreengrassContainer"
            }
        }
    ]
}'
```

Di konsol AWS IoT Greengrass tersebut, Anda dapat menambahkan konektor dari halaman grup Konektor ini. Untuk informasi lebih lanjut, lihat the section called “Memulai dengan konektor (konsol)”.

Data input

Konektor ini menerima metrik pada topik MQTT dan menerbitkan metrik ke. CloudWatch Pesan input harus dalam format JSON.

Filter topik dalam langganan

`cloudwatch/metric/put`

Properti pesan

`request`

Informasi tentang metrik dalam pesan ini.

Objek permintaan berisi data metrik untuk dipublikasikan CloudWatch. Nilai metrik harus memenuhi spesifikasi `PutMetricData`API. Hanya `namespace`, `metricData.metricName`, dan `metricData.value` properti yang diperlukan.

Wajib: `true`

Jenis: `object` yang mencakup properti berikut:

`namespace`

Namespace yang ditentukan pengguna untuk data metrik dalam permintaan ini. CloudWatch menggunakan ruang nama sebagai wadah untuk titik data metrik.

> **Note**
>
> Anda tidak dapat menentukan namespace yang dimulai dengan string yang dicadangkan `AWS/`.

Wajib: `true`

Jenis: `string`

Pola yang valid: `[^:].*`

`metricData`

Data untuk metrik.

Wajib: `true`

Jenis: `object` yang mencakup properti berikut:

`metricName`

Nama metrik.

Wajib: `true`

Jenis: `string`

`dimensions`

Dimensi yang terkait dengan metrik. Dimensi memberikan informasi lebih lanjut tentang metrik dan datanya. Metrik dapat menentukan hingga 10 dimensi.

Konektor ini secara otomatis menyertakan dimensi bernama`coreName`, di mana nilainya adalah nama inti.

Wajib: `false`

Jenis: `array` objek dimensi yang mencakup properti berikut:

`name`

Nama dimensi.

Wajib: `false`

Jenis: `string`

`value`

Nilai dimensi.

Wajib: `false`

Jenis: `string`

`timestamp`

Waktu data metrik diterima, dinyatakan sebagai jumlah detik sejak `Jan 1, 1970 00:00:00 UTC`. Jika nilai ini dihilangkan, konektor menggunakan waktu yang menerima pesan.

Wajib: `false`

Jenis: `timestamp`

> **Note**
>
> Jika Anda menggunakan antara versi 1 dan 4 konektor ini, kami sarankan Anda mengambil stempel waktu secara terpisah untuk setiap metrik saat Anda mengirim beberapa metrik dari satu sumber. Jangan menggunakan variabel untuk menyimpan catatan waktu.

`value`

Nilai untuk metrik.

> **Note**
>
> CloudWatch menolak nilai yang terlalu kecil atau terlalu besar. Nilai harus dalam kisaran `8.515920e-109` ke `1.174271e+108` (Basis 10) atau `2e-360` ke `2e360` (Basis 2). Nilai khusus (sebagai contoh, `NaN`, `+Infinity`, `-Infinity`) tidak didukung.

Wajib: `true`

Jenis: `double`

`unit`

Unit untuk metrik.

Wajib: `false`

Jenis: `string`

Nilai yang valid: `Seconds, Microseconds, Milliseconds, Bytes, Kilobytes, Megabytes, Gigabytes, Terabytes, Bits, Kilobits, Megabits, Gigabits, Terabits, Percent, Count, Bytes/Second, Kilobytes/Second, Megabytes/Second, Gigabytes/Second, Terabytes/Second, Bits/Second, Kilobits/Second, Megabits/Second, Gigabits/Second, Terabits/Second, Count/Second, None`

Batas

Semua batasan yang diberlakukan oleh CloudWatch `PutMetricData`API berlaku untuk metrik saat menggunakan konektor ini. Batas berikut sangat penting:

- Batas 40 KB pada muatan API
- 20 metrik per permintaan API
- 150 transaksi per detik (TPS) untuk API `PutMetricData`

Untuk informasi selengkapnya, lihat CloudWatch batasan di Panduan CloudWatch Pengguna Amazon.

Contoh masukan

```
{
   "request": {
       "namespace": "Greengrass",
       "metricData":
           {
               "metricName": "latency",
               "dimensions": [
                   {
                       "name": "hostname",
                       "value": "test_hostname"
                   }
               ],
               "timestamp": 1539027324,
               "value": 123.0,
```

```
                "unit": "Seconds"
            }
    }
}
```

Data output

Konektor ini menerbitkan informasi status sebagai data output pada topik MQTT.

Filter topik dalam langganan

`cloudwatch/metric/put/status`

Contoh keluaran: Sukses

Responsnya mencakup namespace data metrik dan `RequestId` bidang dari respons. CloudWatch

```
{
   "response": {
        "cloudwatch_rid":"70573243-d723-11e8-b095-75ff2EXAMPLE",
        "namespace": "Greengrass",
        "status":"success"
    }
}
```

Contoh keluaran: Kegagalan

```
{
   "response" : {
        "namespace": "Greengrass",
        "error": "InvalidInputException",
        "error_message":"cw metric is invalid",
        "status":"fail"
   }
}
```

> **Note**
>
> Jika konektor mendeteksi kesalahan yang dapat diulang (sebagai contoh, kesalahan koneksi), konektor mengulang lagi publikasinya dalam batch berikutnya.

Contoh Penggunaan

Gunakan langkah-langkah tingkat tinggi berikut untuk mengatur contoh fungsi Lambda Python 3.7 yang dapat Anda gunakan untuk mencoba konektor.

> **Note**
>
> - Jika Anda menggunakan waktu aktif Python lainnya, Anda dapat membuat symlink dari Python3.x ke Python 3.7.
> - Topik Memulai dengan konektor (konsol) dan Memulai dengan konektor (CLI) berisi langkah-langkah terperinci yang menunjukkan cara mengonfigurasi dan men-deploy contoh konektor Notifikasi Twilio.

1. Pastikan Anda memenuhi persyaratan untuk konektor.

   Untuk persyaratan peran grup, Anda harus mengonfigurasi peran untuk memberikan izin yang diperlukan dan memastikan peran telah ditambahkan ke grup. Untuk informasi lebih lanjut, lihat the section called “Mengelola peran grup (konsol)” atau the section called “Kelola peran grup (CLI)”.
2. Buat dan terbitkan fungsi Lambda yang mengirimkan data input ke konektor.

   Simpan kode contoh sebagai file PY. Unduh dan unzip AWS IoT Greengrass Core SDK for Python. Kemudian, buat paket zip yang berisi file PY dan folder `greengrasssdk` dalam tingkat root. Paket zip ini adalah paket deployment yang Anda unggah ke AWS Lambda.

   Setelah Anda membuat fungsi Lambda Python 3.7, terbitkankan versi fungsi dan buat alias.
3. Konfigurasikan grup Greengrass Anda.

   a. Tambahkan fungsi Lambda dengan aliasnya (direkomendasikan). Konfigurasikan siklus hidup Lambda sebagai berumur panjang (atau `"Pinned": true` dalam CLI).

   b. Tambahkan konektor dan konfigurasikan parameter.

   c. Tambahkan langganan yang mengizinkan konektor untuk menerima data input dan mengirim data output pada filter topik yang didukung.

      - Atur fungsi Lambda sebagai sumber, konektor sebagai target, dan gunakan filter topik input yang mendukung.

- Atur konektor sebagai sumber, AWS IoT Core sebagai target, dan gunakan filter topik input yang mendukung. Anda menggunakan langganan ini untuk melihat pesan status dalam konsol AWS IoT tersebut.

4. Men-deploy grup.
5. Di konsol AWS IoT tersebut, pada halaman Tes ini, berlangganan ke topik data output untuk melihat pesan status dari konektor. Contoh fungsi Lambda yang berumur panjang dan mulai mengirim pesan segera setelah grup dalam-deploy.

   Setelah selesai pengujian, Anda dapat mengatur siklus hidup Lambda ke sesuai permintaan (atau `"Pinned": false` dalam CLI) dan men-deploy grup. Ini menghentikan fungsi dari mengirim pesan.

Contoh

Contoh fungsi Lambda berikut mengirimkan pesan input ke konektor.

```
import greengrasssdk
import time
import json

iot_client = greengrasssdk.client('iot-data')
send_topic = 'cloudwatch/metric/put'

def create_request_with_all_fields():
    return  {
        "request": {
            "namespace": "Greengrass_CW_Connector",
            "metricData": {
                "metricName": "Count1",
                "dimensions": [
                    {
                        "name": "test",
                        "value": "test"
                    }
                ],
                "value": 1,
                "unit": "Seconds",
                "timestamp": time.time()
            }
        }
    }
```

```
def publish_basic_message():
    messageToPublish = create_request_with_all_fields()
    print("Message To Publish: ", messageToPublish)
    iot_client.publish(topic=send_topic,
        payload=json.dumps(messageToPublish))

publish_basic_message()

def lambda_handler(event, context):
    return
```

Lisensi

Konektor CloudWatch Metrik mencakup perangkat lunak/lisensi pihak ketiga berikut:

- AWS SDK for Python (Boto3)/Apache License 2.0
- botocore/Apache License 2.0
- dateutil/PSF License
- docutils/BSD License, Lisensi Publik Umum (GPL) GNU, Lisensi Dasar Perangkat Lunak Python, Domain Publik
- jmespath/MIT License
- s3transfer/Apache License 2.0
- urllib3/Lisensi MIT

Konektor ini dirilis di bawah Perjanjian Lisensi Perangkat lunak core Greengrass.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi konektor.

Versi	Perubahan
5	Perbaiki untuk menambahkan dukungan untuk stempel waktu duplikat dalam data input.

Versi	Perubahan
4	Ditambahkan parameter `IsolationMode` untuk mengonfigurasi mode kontainerisasi untuk konektor.
3	Memperbarui waktu aktif Lambda untuk Python 3.7, yang mengubah persyaratan waktu aktif.
2	Perbaiki untuk mengurangi pencatatan berlebihan.
1	Pelepasan awal.

Sebuah grup Greengrass dapat berisi hanya satu versi konektor pada suatu waktu. Untuk informasi lebih lanjut tentang pembaruan versi konektor, lihat the section called “Versi upgrade konektor”.

Lihat juga

- Integrasikan dengan layanan dan protokol menggunakan konektor
- the section called “Memulai dengan konektor (konsol)”
- the section called “Memulai dengan konektor (CLI)”
- Menggunakan CloudWatch metrik Amazon di CloudWatch Panduan Pengguna Amazon
- PutMetricDatadi Referensi CloudWatch API Amazon

Konektor Pertahanan Perangkat

Pertahanan Perangkat konektor memberitahu administrator perubahan dalam keadaan perangkat core Greengrass. Ini dapat membantu mengidentifikasi perilaku yang tidak biasa yang mungkin menunjukkan perangkat yang dikompromikan.

Konektor ini membaca metrik sistem dari `/proc` perangkat core, dan kemudian menerbitkan metrik untuk AWS IoT Device Defender. Untuk detail pelaporan metrik, lihat Spesifikasi dokumen metrik perangkat dalam AWS IoT Panduan Developer.

Konektor ini memiliki versi berikut.

Versi	ARN
3	`arn:aws:greengrass:region::/connectors/DeviceDefender/versions/3`
2	`arn:aws:greengrass:region::/connectors/DeviceDefender/versions/2`
1	`arn:aws:greengrass:region::/connectors/DeviceDefender/versions/1`

Untuk informasi tentang perubahan versi, lihat Changelog.

Persyaratan

Konektor ini memiliki persyaratan sebagai berikut:

Version 3

- AWS IoT Greengrass perangkat lunak Core v1.9.3 atau yang lebih baru.
- Python versi 3.7 atau 3.8 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.

  > **Note**
  >
  > Untuk menggunakan Python 3.8, jalankan perintah berikut untuk membuat link simbolik dari folder instalasi default Python 3.7 ke binari Python 3.8 yang diinstal.
  >
  > ```
  > sudo ln -s path-to-python-3.8/python3.8 /usr/bin/python3.7
  > ```
  >
  > Ini mengonfigurasi perangkat Anda untuk memenuhi persyaratan Python untuk AWS IoT Greengrass.

- AWS IoT Device Defender dikonfigurasi untuk menggunakan fitur Deteksi untuk melacak pelanggaran. Untuk informasi lebih lanjut, lihat Lacak dalam AWS IoT Panduan Developer.

- Sebuah sumber daya volume lokal dalam grup Greengrass yang menunjuk ke direktori `/proc` ini. Sumber daya harus menggunakan properti berikut:
  - Jalur sumber: `/proc`
  - Jalur tujuan: `/host_proc` (atau nilai yang cocok Pola yang valid)
  - AutoAddGroupOwner: `true`
- Perpustakaan psutil diinstal pada core Greengrass. Versi 5.7.0 adalah versi terkini yang disahkan untuk bekerja dengan konektor.
- Perpustakaan cbor diinstal pada core Greengrass. Versi 1.0.0 adalah versi terbaru yang diverifikasi untuk bekerja dengan konektor.

Versions 1 - 2

- AWS IoT Greengrass perangkat lunak Core v1.7 atau yang lebih baru.
- Python versi 2.7 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.
- AWS IoT Device Defender dikonfigurasi untuk menggunakan fitur Deteksi untuk melacak pelanggaran. Untuk informasi lebih lanjut, lihat Lacak dalam AWS IoT Panduan Developer.
- Sebuah sumber daya volume lokal dalam grup Greengrass yang menunjuk ke direktori `/proc` ini. Sumber daya harus menggunakan properti berikut:
  - Jalur sumber: `/proc`
  - Jalur tujuan: `/host_proc` (atau nilai yang cocok Pola yang valid)
  - AutoAddGroupOwner: `true`
- Perpustakaan psutil diinstal pada core Greengrass.
- Perpustakaan cbor diinstal pada core Greengrass.

Parameter Konektor

Konektor ini menyediakan parameter berikut:

`SampleIntervalSeconds`

Jumlah detik antara setiap siklus pengumpulan dan pelaporan metrik. Nilai minimum adalah 300 detik (5 menit).

Nama tampilan diAWS IoTKonsol: Interval pelaporan metrik

Wajib: `true`

Jenis: `string`

Pola yang valid: `^[0-9]*(?:3[0-9][0-9]|[4-9][0-9]{2}|[1-9][0-9]{3,})$`

`ProcDestinationPath-ResourceId`

ID dari Sumber daya volume `/proc` ini.

> **Note**
>
> Konektor ini diberikan akses baca-saja ke sumber daya.

Nama tampilan diAWS IoTKonsol: Sumber daya untuk direktori/proc

Wajib: `true`

Jenis: `string`

Pola yang valid: `[a-zA-Z0-9_-]+`

`ProcDestinationPath`

Jalur tujuan dari sumber daya volume `/proc` ini.

Nama tampilan diAWS IoTKonsol: Jalur tujuan dari sumber daya /proc

Wajib: `true`

Jenis: `string`

Pola yang valid: `\/[a-zA-Z0-9_-]+`

Buat Contoh Konektor (AWS CLI)

Perintah CLI berikut membuat `ConnectorDefinition` dengan versi awal yang berisi konektor Pertahanan Perangkat.

```
aws greengrass create-connector-definition --name MyGreengrassConnectors --initial-
version '{
    "Connectors": [
        {
            "Id": "MyDeviceDefenderConnector",
```

```
            "ConnectorArn": "arn:aws:greengrass:region::/connectors/DeviceDefender/
versions/3",
            "Parameters": {
                "SampleIntervalSeconds": "600",
                "ProcDestinationPath": "/host_proc",
                "ProcDestinationPath-ResourceId": "my-proc-resource"
            }
        }
    ]
}'
```

> **Note**
>
> Fungsi Lambda dalam konektor mempunyai siklus hidup yang berumur panjang ini.

Di konsol AWS IoT Greengrass tersebut, Anda dapat menambahkan konektor dari halaman grup Konektor ini. Untuk informasi lebih lanjut, lihat the section called “Memulai dengan konektor (konsol)”.

Data input

Konektor ini tidak menerima pesan MQTT sebagai data input.

Data output

Konektor ini menerbitkan metrik keamanan untuk AWS IoT Device Defender sebagai data output.

Filter topik dalam langganan

`$aws/things/+/defender/metrics/json`

> **Note**
>
> Ini adalah sintaks topik yang AWS IoT Device Defender mengharapkan. Konektor menggantikan + wildcard dengan nama perangkat (sebagai contoh, `$aws/things/thing-name/defender/metrics/json`).

Contoh output

Untuk detail pelaporan metrik, lihat Spesifikasi dokumen metrik perangkat dalam AWS IoT Panduan Developer.

```
{
    "header": {
        "report_id": 1529963534,
        "version": "1.0"
    },
    "metrics": {
        "listening_tcp_ports": {
            "ports": [
                {
                    "interface": "eth0",
                    "port": 24800
                },
                {
                    "interface": "eth0",
                    "port": 22
                },
                {
                    "interface": "eth0",
                    "port": 53
                }
            ],
            "total": 3
        },
        "listening_udp_ports": {
            "ports": [
                {
                    "interface": "eth0",
                    "port": 5353
                },
                {
                    "interface": "eth0",
                    "port": 67
                }
            ],
            "total": 2
        },
        "network_stats": {
            "bytes_in": 1157864729406,
            "bytes_out": 1170821865,
            "packets_in": 693092175031,
            "packets_out": 738917180
        },
        "tcp_connections": {
```

```
            "established_connections":{
                "connections": [
                    {
                    "local_interface": "eth0",
                    "local_port": 80,
                    "remote_addr": "192.168.0.1:8000"
                    },
                    {
                    "local_interface": "eth0",
                    "local_port": 80,
                    "remote_addr": "192.168.0.1:8000"
                    }
                ],
                "total": 2
            }
        }
    }
}
```

Lisensi

Konektor ini dirilis di bawah Perjanjian Lisensi Perangkat lunak core Greengrass.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi konektor.

Versi	Perubahan
3	Memperbarui waktu aktif Lambda untuk Python 3.7, yang mengubah persyaratan waktu aktif.
2	Perbaiki untuk mengurangi pencatatan berlebihan.
1	Pelepasan .

Sebuah grup Greengrass dapat berisi hanya satu versi konektor pada suatu waktu. Untuk informasi lebih lanjut tentang pembaruan versi konektor, lihat the section called “Versi upgrade konektor”.

Lihat juga

- Integrasikan dengan layanan dan protokol menggunakan konektor
- the section called “Memulai dengan konektor (konsol)”
- the section called “Memulai dengan konektor (CLI)”
- Pertahanan Perangkat dalam AWS IoT Panduan Developer

Konektor deployment aplikasi docker

Konektor deployment aplikasi Docker Greengrass membuatnya lebih mudah untuk menjalankan gambar Docker Anda pada AWS IoT Greengrass core. Konektor menggunakan Docker Compose untuk memulai aplikasi Docker multi-kontainer dari `docker-compose.yml` file. Khususnya, konektor menjalankan `docker-compose` perintah untuk mengelola kontainer Docker pada perangkat core tunggal. Untuk informasi lebih lanjut, lihat Gambaran umum Docker Compose dalam dokumentasi Docker. Konektor dapat mengakses gambar Docker dipertahankan dalam pendaftar kontainer Docker, seperti Amazon Elastic Container Registry (Amazon ECR), Docker Hub, dan Docker pribadi pendaftar terpercaya.

Setelah Anda menerapkan grup Greengrass, konektor menarik gambar terbaru dan mulai kontainer Docker. Ini menjalankan `docker-compose pull` dan `docker-compose up` perintah. Kemudian, konektor menerbitkan status perintah ke topik MQTT output. Ini juga log informasi status tentang menjalankan kontainer Docker. Ini memungkinkan Anda untuk memantau log aplikasi Anda di Amazon CloudWatch. Untuk informasi selengkapnya, lihat the section called “Pemantauan dengan AWS IoT Greengrass log”. Konektor juga mulai kontainer Docker setiap kali Greengrass daemon restart. Jumlah kontainer Docker yang dapat berjalan pada core tergantung pada hardware Anda.

Kontainer Docker berjalan dalam luar domain Greengrass pada perangkat core, sehingga mereka tidak dapat mengakses inter-process communication (IPC) core. Namun, Anda dapat mengonfigurasi beberapa saluran komunikasi dengan komponen Greengrass, seperti fungsi Lambda lokal. Untuk informasi lebih lanjut, lihat the section called “Berkomunikasi dengan kontainer Docker”.

Anda dapat menggunakan konektor untuk skenario seperti hosting web server atau server MySQL pada perangkat core Anda. Layanan lokal dalam aplikasi Docker Anda dapat berkomunikasi satu sama lain, proses lain dalam lingkungan lokal, dan layanan cloud. Sebagai contoh, Anda dapat menjalankan server web pada core yang mengirimkan permintaan dari fungsi Lambda ke layanan web dalam cloud.

Konektor ini berjalan dalam mode isolasi Tanpa kontainer ini, sehingga Anda dapat men-deploy ke grup Greengrass yang berjalan tanpa kontainerisasi Docker.

Konektor ini memiliki versi berikut.

Versi	ARN
7	`arn:aws:greengrass:region::/connectors/DockerApplicationDeployment/versions/7`
6	`arn:aws:greengrass:region::/connectors/DockerApplicationDeployment/versions/6`
5	`arn:aws:greengrass:region::/connectors/DockerApplicationDeployment/versions/5`
4	`arn:aws:greengrass:region::/connectors/DockerApplicationDeployment/versions/4`
3	`arn:aws:greengrass:region::/connectors/DockerApplicationDeployment/versions/3`
2	`arn:aws:greengrass:region::/connectors/DockerApplicationDeployment/versions/2`
1	`arn:aws:greengrass:region::/connectors/DockerApplicationDeployment/versions/1`

Untuk informasi tentang perubahan versi, lihat Changelog.

Persyaratan

Konektor ini memiliki persyaratan sebagai berikut:

- AWS IoT Greengrass perangkat lunak Core v1.10 atau yang lebih baru.

  > **Note**
  >
  > Konektor ini tidak didukung pada OpenWrt distribusi.

- Python versi 3.7 atau 3.8 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.

  > **Note**
  >
  > Untuk menggunakan Python 3.8, jalankan perintah berikut untuk membuat link simbolik dari folder instalasi default Python 3.7 ke binari Python 3.8 yang diinstal.
  >
  > ```
  > sudo ln -s path-to-python-3.8/python3.8 /usr/bin/python3.7
  > ```
  >
  > Ini mengonfigurasi perangkat Anda untuk memenuhi persyaratan Python untuk AWS IoT Greengrass.

- Minimal 36 MB RAM pada core Greengrass untuk konektor untuk memantau menjalankan kontainer Docker. Kebutuhan memori total tergantung pada jumlah kontainer Docker yang berjalan pada core.
- Mesin Docker 1.9.1 atau yang lebih baru dipasang pada core Greengrass. Versi 19.0.3 adalah versi terkini yang disahkan untuk berfungsi dengan konektor.

  Executable `docker` harus berada dalam direktori `/usr/bin` atau `/usr/local/bin` ini.

  > **Important**
  >
  > Kami sarankan Anda menginstal toko kredensial untuk mengamankan salinan lokal kredensial Docker Anda. Untuk informasi lebih lanjut, lihat the section called “Catatan keamanan”.

Untuk informasi tentang menginstal Docker pada distribusi Amazon Linux, lihat Dasar-dasar Docker untuk Amazon ECS dalam Panduan Pengembang Layanan Kontainer Amazon Elastic.

- Docker Compose dipasang pada core Greengrass. Executable `docker-compose` harus berada dalam direktori `/usr/bin` atau `/usr/local/bin` ini.

  Versi Docker Compose berikut diverifikasi untuk bekerja dengan konektor.

Versi konektor	Versi Docker Diverifikasi
7	1.25.4
6	1.25.4
5	1.25.4
4	1.25.4
3	1.25.4
2	1.25.1
1	1.24.1

- Satu Docker Compose file (sebagai contoh, `docker-compose.yml`), dipertahankan dalam Amazon Simple Storage Service (Amazon S3). Format harus kompatibel dengan versi Docker Compose diinstal pada core. Anda harus menguji file sebelum Anda menggunakannya pada core Anda. Jika Anda mengedit file setelah Anda menerapkan grup Greengrass, Anda harus terapkan ulang grup untuk memperbarui salinan lokal Anda pada core.
- Pengguna Linux dengan izin untuk memanggil daemon Docker lokal dan menulis ke direktori yang menyimpan salinan lokal file Compose Anda. Untuk informasi selengkapnya, lihat Mengatur pengguna Docker pada core.
- Peran grup Greengrass dikonfigurasi untuk mengizinkan `s3:GetObject` tindakan pada bucket S3 yang berisi file Compose Anda. Izin ini ditampilkan dalam kebijakan IAM contoh berikut.

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

```
            "Sid": "AllowAccessToComposeFileS3Bucket",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::bucket-name/*"
        }
    ]
}
```

> **Note**
>
> Jika bucket S3 adalah versioning diaktifkan, maka peran harus dikonfigurasi untuk mengizinkan `s3:GetObjectVersion` tindakan juga. Untuk informasi selengkapnya, lihat Menggunakan versi di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Untuk persyaratan peran grup, Anda harus mengonfigurasi peran untuk memberikan izin yang diperlukan dan memastikan peran telah ditambahkan ke grup. Untuk informasi lebih lanjut, lihat the section called “Mengelola peran grup (konsol)” atau the section called “Kelola peran grup (CLI)”.

- Jika Docker Compose file Anda mereferensikan citra Docker dipertahankan dalam Amazon ECR, peran grup Greengrass dikonfigurasi untuk mengizinkan berikut ini:
  - `ecr:GetDownloadUrlForLayer` dan `ecr:BatchGetImage` tindakan pada repositori Amazon ECR yang berisi gambar Docker.
  - `ecr:GetAuthorizationToken` tindakan pada sumber daya Anda.

  Repositori harus sama Akun AWS dan Wilayah AWS sebagai konektor.

> **Important**
>
> Izin dalam peran grup dapat diasumsikan oleh semua fungsi Lambda dan konektor dalam grup Greengrass. Untuk informasi selengkapnya, lihat the section called “Catatan keamanan”.

Izin-izin ini ditunjukkan pada kebijakan contoh berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowGetEcrRepositories",
            "Effect": "Allow",
            "Action": [
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ],
            "Resource": [
                "arn:aws:ecr:region:account-id:repository/repository-name"
            ]
        },
        {
            "Sid": "AllowGetEcrAuthToken",
            "Effect": "Allow",
            "Action": "ecr:GetAuthorizationToken",
            "Resource": "*"
        }
    ]
}
```

Untuk informasi lebih lanjut, lihat Contoh kebijakan repositori Amazon ECR dalam Panduan Pengguna Amazon ECR.

Untuk persyaratan peran grup, Anda harus mengonfigurasi peran untuk memberikan izin yang diperlukan dan memastikan peran telah ditambahkan ke grup. Untuk informasi lebih lanjut, lihat the section called “Mengelola peran grup (konsol)” atau the section called “Kelola peran grup (CLI)”.

- Jika file Docker Compose Anda mereferensikan gambar Docker dari AWS Marketplace, konektor juga memiliki persyaratan sebagai berikut:
  - Anda harus berlangganan AWS Marketplace produk kontainer. Untuk informasi lebih lanjut, lihat Menemukan dan berlangganan produk kontainer dalam AWS Marketplace Panduan Pelanggan.
  - AWS IoT Greengrass harus dikonfigurasi untuk mendukung rahasia lokal, seperti yang dijelaskan dalam Persyaratan Rahasia. Konektor menggunakan fitur ini hanya untuk mengambil rahasia Anda dari AWS Secrets Manager, bukan untuk menyimpannya.
  - Anda harus membuat rahasia dalam Secrets Manager untuk setiap registri AWS Marketplace yang menyimpan citra Docker direferensikan dalam file Compose Anda. Untuk informasi selengkapnya, lihat the section called “Mengakses gambar Docker dari repositori pribadi”.

- Jika file Docker Compose Anda mereferensikan gambar Docker dari repositori pribadi dalam pendaftar selain Amazon ECR, seperti Docker Hub, konektor juga memiliki persyaratan sebagai berikut:
  - AWS IoT Greengrass harus dikonfigurasi untuk mendukung rahasia lokal, seperti yang dijelaskan dalam Persyaratan Rahasia. Konektor menggunakan fitur ini hanya untuk mengambil rahasia Anda dari AWS Secrets Manager, bukan untuk menyimpannya.
  - Anda harus membuat rahasia dalam Secrets Manager untuk setiap repositori pribadi yang menyimpan gambar Docker direferensikan dalam file Compose Anda. Untuk informasi selengkapnya, lihat the section called “Mengakses gambar Docker dari repositori pribadi”.
- Docker daemon harus berjalan ketika Anda menerapkan grup Greengrass yang berisi konektor ini.

Mengakses gambar Docker dari repositori pribadi

Jika Anda menggunakan kredensial untuk mengakses gambar Docker Anda, maka Anda harus mengizinkan konektor untuk mengaksesnya. Cara Anda melakukannya tergantung dalam mana gambar Docker berada.

Untuk gambar Docker dipertahankan Amazon ECR, Anda memberikan izin untuk mendapatkan token otorisasi Anda dalam peran grup Greengrass. Untuk informasi selengkapnya, lihat the section called “Persyaratan”.

Untuk gambar Docker yang dipertahankan dalam repositori atau pendaftar pribadi lainnya, Anda harus membuat rahasia dalam AWS Secrets Manager untuk menyimpan informasi login Anda. Ini termasuk gambar Docker yang Anda langgani dalam AWS Marketplace. Buat satu rahasia untuk setiap repositori. Jika Anda memperbarui rahasia Anda dalam Secrets Manager, perubahan menyebar ke core waktu berikutnya bahwa Anda men-deploy grup.

> **Note**
>
> Secrets Manager adalah layanan yang dapat Anda gunakan untuk aman menyimpan dan mengelola kredensial, kunci, dan rahasia lainnya dalam AWS Cloud. Untuk informasi lebih lanjut, lihat Apa AWS Secrets Manager? dalam AWS Secrets Manager Panduan Pengguna.

Setiap rahasia harus berisi kunci berikut:

Kunci	Nilai
`username`	Nama pengguna yang digunakan untuk mengakses repositori atau registri.
`password`	Kata sandi yang digunakan untuk mengakses repositori atau registri.
`registryUrl`	Titik akhir registri. Ini harus cocok dengan URL registri yang sesuai dalam file Compose.

> **Note**
>
> Mengizinkan AWS IoT Greengrass untuk mengakses rahasia secara default, nama rahasia harus dimulai dengan Greengrass-. Jika tidak, peran layanan Greengrass Anda harus memberikan akses. Untuk informasi selengkapnya, lihat the section called “Mengizinkan AWS IoT Greengrass untuk mendapatkan nilai rahasia”.

Untuk mendapatkan informasi login untuk gambar Docker dari AWS Marketplace

1. Dapatkan kata sandi untuk gambar Docker darid AWS Marketplace dengan menggunakan `aws ecr get-login-password` perintah. Untuk informasi selengkapnya, lihat get-login-password dalam AWS CLI Referensi Perintah.

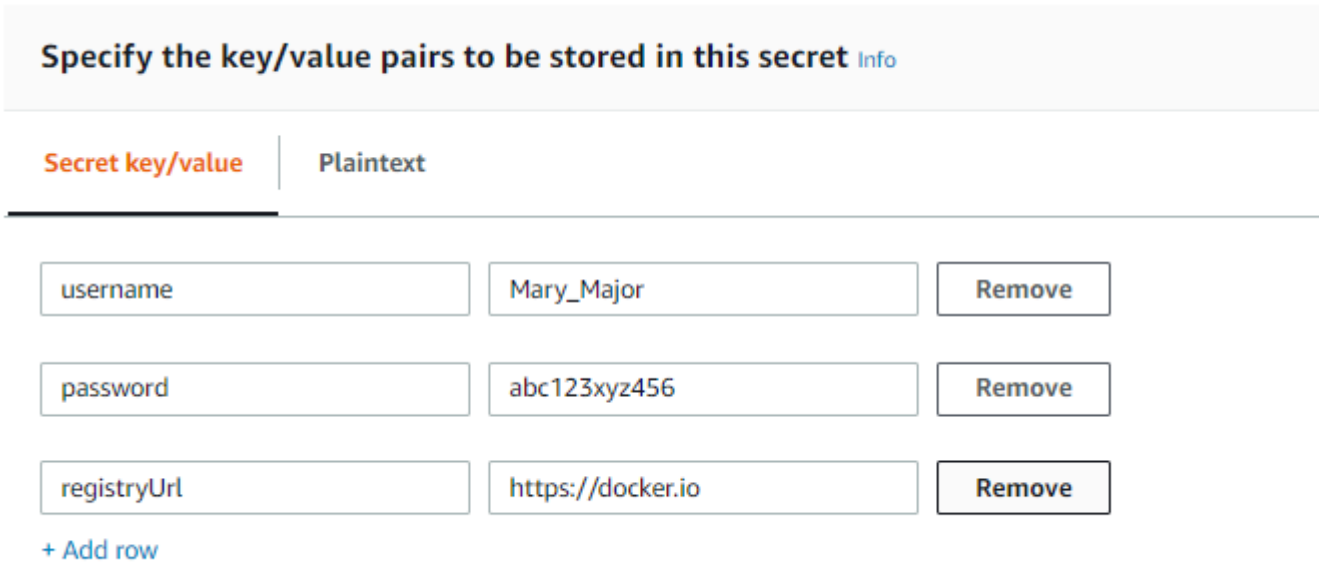
   ```
   aws ecr get-login-password
   ```

2. Mengambil URL registri untuk gambar Docker. Buka situs AWS Marketplace web, dan buka halaman peluncuran produk kontainer. Di bawah Citra Kontainer, pilih Melihat detail gambar kontainer untuk menemukan nama pengguna dan URL registri.

Gunakan nama pengguna, sandi, dan URL registri yang diambil untuk membuat rahasia untuk setiap AWS Marketplace registri yang menyimpan gambar Docker direferensikan dalam file Compose Anda.

Untuk membuat rahasia (konsol)

Di AWS Secrets Manager konsol, pilih Jenis rahasia lainnya. Di bawah Tentukan pasangan nilai kunci yang akan dipertahankan untuk rahasia ini, tambahkan baris untuk `username`, `password`, dan `registryUrl`. Untuk informasi lebih lanjut, lihat Membuat rahasia dasar dalam AWS Secrets Manager Panduan Pengguna.



Untuk membuat rahasia (CLI)

Di AWS CLI, gunakan perintah `create-secret` Secrets Manager, seperti yang ditunjukkan dalam contoh berikut. Untuk informasi lebih lanjut, lihat buat-bucket dalam AWS CLI Referensi Perintah.

```
aws secretsmanager create-secret --name greengrass-MySecret --secret-string
 [{"username":"Mary_Major"},{"password":"abc123xyz456"},{"registryUrl":"https://
docker.io"}]
```

Important

Ini adalah tanggung jawab Anda untuk mengamankan `DockerComposeFileDestinationPath` direktori yang menyimpan file Docker Compose dan kredensial untuk gambar Docker Anda dari repositori pribadi. Untuk informasi selengkapnya, lihat the section called “Catatan keamanan”.

Parameter

Konektor ini menyediakan parameter berikut:

Version 7

`DockerComposeFileS3Bucket`

Nama bucket S3 yang berisi file Docker Compose Anda. Saat membuat bucket, pastikan untuk mengikuti aturan nama bucket yang dijelaskan dalam Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Nama tampilan pada konsol AWS IoT tersebut: File Docker Compose dalam S3

> **Note**
>
> Di konsol tersebut, properti file Docker Compose dalam S3 menggabungkan parameter `DockerComposeFileS3Bucket`, `DockerComposeFileS3Key`, dan `DockerComposeFileS3Version` ini.

Wajib: `true`

Jenis: `string`

Pola yang valid `[a-zA-Z0-9\\-\\.]{3,63}`

`DockerComposeFileS3Key`

Kunci objek untuk file Docker Compose Anda dalam Amazon S3. Untuk informasi selengkapnya, termasuk pedoman penamaan kunci objek, lihat Kunci objek dan metadata di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

> **Note**
>
> Di konsol tersebut, properti file Docker Compose dalam S3 menggabungkan parameter `DockerComposeFileS3Bucket`, `DockerComposeFileS3Key`, dan `DockerComposeFileS3Version` ini.

Wajib: `true`

Jenis: `string`

Pola yang valid `.+`

`DockerComposeFileS3Version`

Versi objek untuk file Docker Compose Anda dalam Amazon S3. Untuk informasi selengkapnya, termasuk pedoman penamaan kunci objek, lihat Menggunakan versi di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

> **Note**
>
> Di konsol tersebut, properti file Docker Compose dalam S3 menggabungkan parameter `DockerComposeFileS3Bucket`, `DockerComposeFileS3Key`, dan `DockerComposeFileS3Version` ini.

Wajib: `false`

Jenis: `string`

Pola yang valid `.+`

`DockerComposeFileDestinationPath`

Jalur absolut dari direktori lokal yang digunakan untuk menyimpan salinan file Docker Compose. Ini harus berupa direktori yang sudah ada. Pengguna yang ditentukan untuk `DockerUserId` harus memiliki izin untuk membuat file dalam direktori ini. Untuk informasi selengkapnya, lihat the section called “Mengatur pengguna Docker pada core”.

> **Important**
>
> Direktori ini menyimpan file Docker Compose Anda dan kredensial untuk citra Docker Anda dari repositori privat. Ini adalah tanggung jawab Anda untuk mengamankan direktori ini. Untuk informasi selengkapnya, lihat the section called “Catatan keamanan”.

Nama tampilan pada konsol AWS IoT tersebut: Jalur direktori untuk file Compose lokal

Wajib: `true`

Jenis: `string`

Pola yang valid `\/.*\/?`

Contoh: `/home/username/myCompose`

`DockerUserId`

UID dari pengguna Linux yang konektor berjalan sebagai. Pengguna ini harus milik grup Linux `docker` pada perangkat core dan memiliki izin menulis ke direktori `DockerComposeFileDestinationPath` ini. Untuk informasi selengkapnya, lihat Mengatur pengguna Docker pada core.

> **Note**
>
> Kami merekomendasikan bahwa Anda menghindari berjalan sebagai root kecuali benar-benar diperlukan. Jika Anda menentukan pengguna root, Anda harus mengizinkan fungsi Lambda untuk menjalankan sebagai root pada AWS IoT Greengrass core. Untuk informasi selengkapnya, lihat the section called “Menjalankan fungsi Lambda sebagai root”.

Nama tampilan pada konsol AWS IoT tersebut: ID pengguna Docker

Wajib: `false`

Jenis: `string`

Pola yang valid: `^[0-9]{1,5}$`

`AWSSecretsArnList`

Amazon Resource Names (ARN) dari rahasia dalam AWS Secrets Manager yang berisi informasi login yang digunakan untuk mengakses gambar Docker Anda dalam repositori pribadi. Untuk informasi selengkapnya, lihat the section called “Mengakses gambar Docker dari repositori pribadi”.

Nama tampilan pada konsol AWS IoT tersebut: Kredensial untuk repositori privat

Diperlukan:`false`. Parameter ini diperlukan untuk mengakses citra Docker yang dipertahankan dalam repositori privat.

Jenis: `array` dari `string`

Pola yang valid: `[( ?,? ?"(arn:(aws(-[a-z]+)):secretsmanager:[a-z0-9-]+:[0-9]{12}:secret:([a-zA-Z0-9\]+/)[a-zA-Z0-9/_+=,.@-]+-[a-zA-Z0-9]+)")]`

`DockerContainerStatusLogFrequency`

Frekuensi (dalam detik) saat konektor mencatat informasi status tentang kontainer Docker yang berjalan pada core. Default-nya adalah 300 detik (5 menit).

Nama tampilan pada konsol AWS IoT tersebut: Frekuensi pencatatan

Wajib: `false`

Jenis: `string`

Pola yang valid: `^[1-9]{1}[0-9]{0,3}$`

`ForceDeploy`

Mengindikasikan apakah akan memaksa deployment Docker jika gagal karena pembersihan yang tidak benar dari deployment terakhir. Nilai default-nya adalah `False`.

Nama tampilan pada konsol AWS IoT tersebut: deployment paksa

Wajib: `false`

Jenis: `string`

Pola yang valid: `^(true|false)$`

`DockerPullBeforeUp`

Menunjukkan apakah deployer harus berjalan `docker-compose pull` sebelum menjalankan `docker-compose up` pull-down-up perilaku. Nilai default-nya adalah `True`.

Nama tampilan pada konsol AWS IoT tersebut: Docker Tarik Sebelum Naik

Wajib: `false`

Jenis: `string`

Pola yang valid: `^(true|false)$`

`StopContainersOnNewDeployment`

Mengindikasikan apakah konektor harus menghentikan kontainer Docker Deployer managed docker ketika GGC dihentikan (GGC berhenti ketika grup baru diterapkan, atau kernel dimatikan). Nilai default-nya adalah `True`.

Nama tampilan pada konsol AWS IoT tersebut: Docker berhenti pada deployment baru

> **Note**
>
> Kami menyarankan agar parameter ini diatur ke default `True` nilai. Parameter untuk `False` menyebabkan kontainer Docker Anda terus berjalan bahkan setelah mengakhiri AWS IoT Greengrass core atau memulai deployment baru. Jika Anda mengatur parameter ini ke `False`, Anda harus memastikan bahwa kontainer Docker Anda dipertahankan seperlunya dalam hal `docker-compose` perubahan nama layanan atau penambahan.
> Untuk informasi lebih lanjut, lihat `docker-compose` dokumentasi file compose.

Wajib: `false`

Jenis: `string`

Pola yang valid: `^(true|false)$`

`DockerOfflineMode`

Mengindikasikan apakah akan menggunakan file Docker Compose yang ada ketika AWS IoT Greengrass mulai offline. Nilai default-nya adalah `False`.

Wajib: `false`

Jenis: `string`

Pola yang valid: `^(true|false)$`

Version 6

`DockerComposeFileS3Bucket`

Nama bucket S3 yang berisi file Docker Compose Anda. Saat membuat bucket, pastikan untuk mengikuti aturan nama bucket yang dijelaskan dalam Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Nama tampilan pada konsol AWS IoT tersebut: File Docker Compose dalam S3

> **Note**
>
> Di konsol tersebut, properti file Docker Compose dalam S3 menggabungkan parameter `DockerComposeFileS3Bucket`, `DockerComposeFileS3Key`, dan `DockerComposeFileS3Version` ini.

Wajib: `true`

Jenis: `string`

Pola yang valid `[a-zA-Z0-9\\-\\.]{3,63}`

`DockerComposeFileS3Key`

Kunci objek untuk file Docker Compose Anda dalam Amazon S3. Untuk informasi selengkapnya, termasuk pedoman penamaan kunci objek, lihat Kunci objek dan metadata di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

> **Note**
>
> Di konsol tersebut, properti file Docker Compose dalam S3 menggabungkan parameter `DockerComposeFileS3Bucket`, `DockerComposeFileS3Key`, dan `DockerComposeFileS3Version` ini.

Wajib: `true`

Jenis: `string`

Pola yang valid `.+`

`DockerComposeFileS3Version`

Versi objek untuk file Docker Compose Anda dalam Amazon S3. Untuk informasi selengkapnya, termasuk pedoman penamaan kunci objek, lihat Menggunakan versi di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

> **Note**
>
> Di konsol tersebut, properti file Docker Compose dalam S3 menggabungkan parameter `DockerComposeFileS3Bucket`, `DockerComposeFileS3Key`, dan `DockerComposeFileS3Version` ini.

Wajib: `false`

Jenis: `string`

Pola yang valid `.+`

`DockerComposeFileDestinationPath`

Jalur absolut dari direktori lokal yang digunakan untuk menyimpan salinan file Docker Compose. Ini harus berupa direktori yang sudah ada. Pengguna yang ditentukan untuk `DockerUserId` harus memiliki izin untuk membuat file dalam direktori ini. Untuk informasi selengkapnya, lihat the section called “Mengatur pengguna Docker pada core”.

> **Important**
>
> Direktori ini menyimpan file Docker Compose Anda dan kredensial untuk citra Docker Anda dari repositori privat. Ini adalah tanggung jawab Anda untuk mengamankan direktori ini. Untuk informasi selengkapnya, lihat the section called “Catatan keamanan”.

Nama tampilan pada konsol AWS IoT tersebut: Jalur direktori untuk file Compose lokal

Wajib: `true`

Jenis: `string`

Pola yang valid `\/.*\/?`

Contoh: `/home/username/myCompose`

`DockerUserId`

UID dari pengguna Linux yang konektor berjalan sebagai. Pengguna ini harus milik grup Linux `docker` pada perangkat core dan memiliki izin menulis ke direktori `DockerComposeFileDestinationPath` ini. Untuk informasi selengkapnya, lihat Mengatur pengguna Docker pada core.

> **Note**
>
> Kami merekomendasikan bahwa Anda menghindari berjalan sebagai root kecuali benar-benar diperlukan. Jika Anda menentukan pengguna root, Anda harus mengizinkan fungsi Lambda untuk menjalankan sebagai root pada AWS IoT Greengrass core. Untuk informasi selengkapnya, lihat the section called “Menjalankan fungsi Lambda sebagai root”.

Nama tampilan pada konsol AWS IoT tersebut: ID pengguna Docker

Wajib: `false`

Jenis: `string`

Pola yang valid: `^[0-9]{1,5}$`

`AWSSecretsArnList`

Amazon Resource Names (ARN) dari rahasia dalam AWS Secrets Manager yang berisi informasi login yang digunakan untuk mengakses gambar Docker Anda dalam repositori pribadi. Untuk informasi selengkapnya, lihat the section called “Mengakses gambar Docker dari repositori pribadi”.

Nama tampilan pada konsol AWS IoT tersebut: Kredensial untuk repositori privat

Diperlukan:`false`. Parameter ini diperlukan untuk mengakses citra Docker yang dipertahankan dalam repositori privat.

Jenis: `array` dari `string`

Pola yang valid: `[( ?,? ?"(arn:(aws(-[a-z]+)):secretsmanager:[a-z0-9-]+:[0-9]{12}:secret:([a-zA-Z0-9\]+/)[a-zA-Z0-9/_+=,.@-]+-[a-zA-Z0-9]+)")]`

`DockerContainerStatusLogFrequency`

Frekuensi (dalam detik) saat konektor mencatat informasi status tentang kontainer Docker yang berjalan pada core. Default-nya adalah 300 detik (5 menit).

Nama tampilan pada konsol AWS IoT tersebut: Frekuensi pencatatan

Wajib: `false`

Jenis: `string`

Pola yang valid: `^[1-9]{1}[0-9]{0,3}$`

`ForceDeploy`

Mengindikasikan apakah akan memaksa deployment Docker jika gagal karena pembersihan yang tidak benar dari deployment terakhir. Nilai default-nya adalah `False`.

Nama tampilan pada konsol AWS IoT tersebut: deployment paksa

Wajib: `false`

Jenis: `string`

Pola yang valid: `^(true|false)$`

`DockerPullBeforeUp`

Menunjukkan apakah deployer harus berjalan `docker-compose pull` sebelum menjalankan `docker-compose up` pull-down-up perilaku. Nilai default-nya adalah `True`.

Nama tampilan pada konsol AWS IoT tersebut: Docker Tarik Sebelum Naik

Wajib: `false`

Jenis: `string`

Pola yang valid: `^(true|false)$`

`StopContainersOnNewDeployment`

Mengindikasikan apakah konektor harus menghentikan Docker Deployer dikelola kontainer docker ketika GGC dihentikan (ketika deployment grup baru dibuat, atau kernel dimatikan). Nilai default-nya adalah `True`.

Nama tampilan pada konsol AWS IoT tersebut: Docker berhenti pada deployment baru

> **Note**
>
> Kami menyarankan agar parameter ini diatur ke default `True` nilai. Parameter untuk `False` menyebabkan kontainer Docker Anda terus berjalan bahkan setelah mengakhiri AWS IoT Greengrass core atau memulai deployment baru. Jika Anda mengatur parameter ini ke `False`, Anda harus memastikan bahwa kontainer Docker Anda dipertahankan seperlunya dalam hal `docker-compose` perubahan nama layanan atau penambahan.
> Untuk informasi lebih lanjut, lihat `docker-compose` dokumentasi file compose.

Wajib: `false`

Jenis: `string`

Pola yang valid: `^(true|false)$`

Version 5

`DockerComposeFileS3Bucket`

Nama bucket S3 yang berisi file Docker Compose Anda. Saat membuat bucket, pastikan untuk mengikuti aturan nama bucket yang dijelaskan dalam Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Nama tampilan pada konsol AWS IoT tersebut: File Docker Compose dalam S3

> **Note**
>
> Di konsol tersebut, properti file Docker Compose dalam S3 menggabungkan parameter `DockerComposeFileS3Bucket`, `DockerComposeFileS3Key`, dan `DockerComposeFileS3Version` ini.

Wajib: `true`

Jenis: `string`

Pola yang valid `[a-zA-Z0-9\\-\\.]{3,63}`

`DockerComposeFileS3Key`

Kunci objek untuk file Docker Compose Anda dalam Amazon S3. Untuk informasi selengkapnya, termasuk pedoman penamaan kunci objek, lihat Kunci objek dan metadata di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

> **Note**
>
> Di konsol tersebut, properti file Docker Compose dalam S3 menggabungkan parameter `DockerComposeFileS3Bucket`, `DockerComposeFileS3Key`, dan `DockerComposeFileS3Version` ini.

Wajib: `true`

Jenis: `string`

Pola yang valid `.+`

`DockerComposeFileS3Version`

Versi objek untuk file Docker Compose Anda dalam Amazon S3. Untuk informasi selengkapnya, termasuk pedoman penamaan kunci objek, lihat Menggunakan versi di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

> **Note**
>
> Di konsol tersebut, properti file Docker Compose dalam S3 menggabungkan parameter `DockerComposeFileS3Bucket`, `DockerComposeFileS3Key`, dan `DockerComposeFileS3Version` ini.

Wajib: `false`

Jenis: `string`

Pola yang valid `.+`

`DockerComposeFileDestinationPath`

Jalur absolut dari direktori lokal yang digunakan untuk menyimpan salinan file Docker Compose. Ini harus berupa direktori yang sudah ada. Pengguna yang ditentukan untuk `DockerUserId` harus memiliki izin untuk membuat file dalam direktori ini. Untuk informasi selengkapnya, lihat the section called “Mengatur pengguna Docker pada core”.

> **Important**
>
> Direktori ini menyimpan file Docker Compose Anda dan kredensial untuk citra Docker Anda dari repositori privat. Ini adalah tanggung jawab Anda untuk mengamankan direktori ini. Untuk informasi selengkapnya, lihat the section called “Catatan keamanan”.

Nama tampilan pada konsol AWS IoT tersebut: Jalur direktori untuk file Compose lokal

Wajib: `true`

Jenis: `string`

Pola yang valid `\/.*\/?`

Contoh: `/home/username/myCompose`

`DockerUserId`

UID dari pengguna Linux yang konektor berjalan sebagai. Pengguna ini harus milik grup Linux `docker` pada perangkat core dan memiliki izin menulis ke direktori `DockerComposeFileDestinationPath` ini. Untuk informasi selengkapnya, lihat Mengatur pengguna Docker pada core.

> **Note**
>
> Kami merekomendasikan bahwa Anda menghindari berjalan sebagai root kecuali benar-benar diperlukan. Jika Anda menentukan pengguna root, Anda harus mengizinkan fungsi Lambda untuk menjalankan sebagai root pada AWS IoT Greengrass core. Untuk informasi selengkapnya, lihat the section called “Menjalankan fungsi Lambda sebagai root”.

Nama tampilan pada konsol AWS IoT tersebut: ID pengguna Docker

Wajib: `false`

Jenis: `string`

Pola yang valid: `^[0-9]{1,5}$`

`AWSSecretsArnList`

Amazon Resource Names (ARN) dari rahasia dalam AWS Secrets Manager yang berisi informasi login yang digunakan untuk mengakses gambar Docker Anda dalam repositori pribadi. Untuk informasi selengkapnya, lihat the section called “Mengakses gambar Docker dari repositori pribadi”.

Nama tampilan pada konsol AWS IoT tersebut: Kredensial untuk repositori privat

Diperlukan:`false`. Parameter ini diperlukan untuk mengakses citra Docker yang dipertahankan dalam repositori privat.

Jenis: `array` dari `string`

Pola yang valid: `[( ?,? ?"(arn:(aws(-[a-z]+)):secretsmanager:[a-z0-9-]+:[0-9]{12}:secret:([a-zA-Z0-9\]+/)[a-zA-Z0-9/_+=,.@-]+-[a-zA-Z0-9]+)")]`

`DockerContainerStatusLogFrequency`

Frekuensi (dalam detik) saat konektor mencatat informasi status tentang kontainer Docker yang berjalan pada core. Default-nya adalah 300 detik (5 menit).

Nama tampilan pada konsol AWS IoT tersebut: Frekuensi pencatatan

Wajib: `false`

Jenis: `string`

Pola yang valid: `^[1-9]{1}[0-9]{0,3}$`

`ForceDeploy`

Mengindikasikan apakah akan memaksa deployment Docker jika gagal karena pembersihan yang tidak benar dari deployment terakhir. Nilai default-nya adalah `False`.

Nama tampilan pada konsol AWS IoT tersebut: deployment paksa

Wajib: `false`

Jenis: `string`

Pola yang valid: `^(true|false)$`

`DockerPullBeforeUp`

Menunjukkan apakah deployer harus berjalan `docker-compose pull` sebelum menjalankan `docker-compose up` pull-down-up perilaku. Nilai default-nya adalah `True`.

Nama tampilan pada konsol AWS IoT tersebut: Docker Tarik Sebelum Naik

Wajib: `false`

Jenis: `string`

Pola yang valid: `^(true|false)$`

Versions 2 - 4

`DockerComposeFileS3Bucket`

Nama bucket S3 yang berisi file Docker Compose Anda. Saat membuat bucket, pastikan untuk mengikuti aturan nama bucket yang dijelaskan dalam Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Nama tampilan pada konsol AWS IoT tersebut: File Docker Compose dalam S3

> **Note**
>
> Di konsol tersebut, properti file Docker Compose dalam S3 menggabungkan parameter `DockerComposeFileS3Bucket`, `DockerComposeFileS3Key`, dan `DockerComposeFileS3Version` ini.

Wajib: `true`

Jenis: `string`

Pola yang valid `[a-zA-Z0-9\\-\\.]{3,63}`

`DockerComposeFileS3Key`

Kunci objek untuk file Docker Compose Anda dalam Amazon S3. Untuk informasi selengkapnya, termasuk pedoman penamaan kunci objek, lihat Kunci objek dan metadata di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

> **Note**
>
> Di konsol tersebut, properti file Docker Compose dalam S3 menggabungkan parameter `DockerComposeFileS3Bucket`, `DockerComposeFileS3Key`, dan `DockerComposeFileS3Version` ini.

Wajib: `true`

Jenis: `string`

Pola yang valid `.+`

`DockerComposeFileS3Version`

Versi objek untuk file Docker Compose Anda dalam Amazon S3. Untuk informasi selengkapnya, termasuk pedoman penamaan kunci objek, lihat Menggunakan versi di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

> **Note**
>
> Di konsol tersebut, properti file Docker Compose dalam S3 menggabungkan parameter `DockerComposeFileS3Bucket`, `DockerComposeFileS3Key`, dan `DockerComposeFileS3Version` ini.

Wajib: `false`

Jenis: `string`

Pola yang valid `.+`

`DockerComposeFileDestinationPath`

Jalur absolut dari direktori lokal yang digunakan untuk menyimpan salinan file Docker Compose. Ini harus berupa direktori yang sudah ada. Pengguna yang ditentukan untuk

`DockerUserId` harus memiliki izin untuk membuat file dalam direktori ini. Untuk informasi selengkapnya, lihat the section called “Mengatur pengguna Docker pada core”.

> **Important**
>
> Direktori ini menyimpan file Docker Compose Anda dan kredensial untuk citra Docker Anda dari repositori privat. Ini adalah tanggung jawab Anda untuk mengamankan direktori ini. Untuk informasi selengkapnya, lihat the section called “Catatan keamanan”.

Nama tampilan pada konsol AWS IoT tersebut: Jalur direktori untuk file Compose lokal

Wajib: `true`

Jenis: `string`

Pola yang valid `\/.*\/?`

Contoh: `/home/username/myCompose`

`DockerUserId`

UID dari pengguna Linux yang konektor berjalan sebagai. Pengguna ini harus milik grup Linux `docker` pada perangkat core dan memiliki izin menulis ke direktori `DockerComposeFileDestinationPath` ini. Untuk informasi selengkapnya, lihat Mengatur pengguna Docker pada core.

> **Note**
>
> Kami merekomendasikan bahwa Anda menghindari berjalan sebagai root kecuali benar-benar diperlukan. Jika Anda menentukan pengguna root, Anda harus mengizinkan fungsi Lambda untuk menjalankan sebagai root pada AWS IoT Greengrass core. Untuk informasi selengkapnya, lihat the section called “Menjalankan fungsi Lambda sebagai root”.

Nama tampilan pada konsol AWS IoT tersebut: ID pengguna Docker

Wajib: `false`

Jenis: `string`

Pola yang valid: `^[0-9]{1,5}$`

`AWSSecretsArnList`

Amazon Resource Names (ARN) dari rahasia dalam AWS Secrets Manager yang berisi informasi login yang digunakan untuk mengakses gambar Docker Anda dalam repositori pribadi. Untuk informasi selengkapnya, lihat the section called “Mengakses gambar Docker dari repositori pribadi”.

Nama tampilan pada konsol AWS IoT tersebut: Kredensial untuk repositori privat

Diperlukan:`false`. Parameter ini diperlukan untuk mengakses citra Docker yang dipertahankan dalam repositori privat.

Jenis: `array` dari `string`

Pola yang valid: `[( ?,? ?"(arn:(aws(-[a-z]+)):secretsmanager:[a-z0-9-]+:[0-9]{12}:secret:([a-zA-Z0-9\]+/)[a-zA-Z0-9/_+=,.@-]+-[a-zA-Z0-9]+)")]`

`DockerContainerStatusLogFrequency`

Frekuensi (dalam detik) saat konektor mencatat informasi status tentang kontainer Docker yang berjalan pada core. Default-nya adalah 300 detik (5 menit).

Nama tampilan pada konsol AWS IoT tersebut: Frekuensi pencatatan

Wajib: `false`

Jenis: `string`

Pola yang valid: `^[1-9]{1}[0-9]{0,3}$`

`ForceDeploy`

Mengindikasikan apakah akan memaksa deployment Docker jika gagal karena pembersihan yang tidak benar dari deployment terakhir. Nilai default-nya adalah `False`.

Nama tampilan pada konsol AWS IoT tersebut: deployment paksa

Wajib: `false`

Jenis: `string`

Pola yang valid: `^(true|false)$`

Version 1

`DockerComposeFileS3Bucket`

Nama bucket S3 yang berisi file Docker Compose Anda. Saat membuat bucket, pastikan untuk mengikuti aturan nama bucket yang dijelaskan dalam Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Nama tampilan pada konsol AWS IoT tersebut: File Docker Compose dalam S3

> **Note**
>
> Di konsol tersebut, properti file Docker Compose dalam S3 menggabungkan parameter `DockerComposeFileS3Bucket`, `DockerComposeFileS3Key`, dan `DockerComposeFileS3Version` ini.

Wajib: `true`

Jenis: `string`

Pola yang valid `[a-zA-Z0-9\\-\\.]{3,63}`

`DockerComposeFileS3Key`

Kunci objek untuk file Docker Compose Anda dalam Amazon S3. Untuk informasi selengkapnya, termasuk pedoman penamaan kunci objek, lihat Kunci objek dan metadata di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

> **Note**
>
> Di konsol tersebut, properti file Docker Compose dalam S3 menggabungkan parameter `DockerComposeFileS3Bucket`, `DockerComposeFileS3Key`, dan `DockerComposeFileS3Version` ini.

Wajib: `true`

Jenis: `string`

Pola yang valid `.+`

`DockerComposeFileS3Version`

Versi objek untuk file Docker Compose Anda dalam Amazon S3. Untuk informasi selengkapnya, termasuk pedoman penamaan kunci objek, lihat Menggunakan versi di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

> **Note**
>
> Di konsol tersebut, properti file Docker Compose dalam S3 menggabungkan parameter `DockerComposeFileS3Bucket`, `DockerComposeFileS3Key`, dan `DockerComposeFileS3Version` ini.

Wajib: `false`

Jenis: `string`

Pola yang valid `.+`

`DockerComposeFileDestinationPath`

Jalur absolut dari direktori lokal yang digunakan untuk menyimpan salinan file Docker Compose. Ini harus berupa direktori yang sudah ada. Pengguna yang ditentukan untuk `DockerUserId` harus memiliki izin untuk membuat file dalam direktori ini. Untuk informasi selengkapnya, lihat the section called “Mengatur pengguna Docker pada core”.

> **Important**
>
> Direktori ini menyimpan file Docker Compose Anda dan kredensial untuk citra Docker Anda dari repositori privat. Ini adalah tanggung jawab Anda untuk mengamankan direktori ini. Untuk informasi selengkapnya, lihat the section called “Catatan keamanan”.

Nama tampilan pada konsol AWS IoT tersebut: Jalur direktori untuk file Compose lokal

Wajib: `true`

Jenis: `string`

Pola yang valid `\/.*\/?`

Contoh: `/home/username/myCompose`

`DockerUserId`

UID dari pengguna Linux yang konektor berjalan sebagai. Pengguna ini harus milik grup Linux `docker` pada perangkat core dan memiliki izin menulis ke direktori `DockerComposeFileDestinationPath` ini. Untuk informasi selengkapnya, lihat Mengatur pengguna Docker pada core.

> **Note**
>
> Kami merekomendasikan bahwa Anda menghindari berjalan sebagai root kecuali benar-benar diperlukan. Jika Anda menentukan pengguna root, Anda harus mengizinkan fungsi Lambda untuk menjalankan sebagai root pada AWS IoT Greengrass core. Untuk informasi selengkapnya, lihat the section called “Menjalankan fungsi Lambda sebagai root”.

Nama tampilan pada konsol AWS IoT tersebut: ID pengguna Docker

Wajib: `false`

Jenis: `string`

Pola yang valid: `^[0-9]{1,5}$`

`AWSSecretsArnList`

Amazon Resource Names (ARN) dari rahasia dalam AWS Secrets Manager yang berisi informasi login yang digunakan untuk mengakses gambar Docker Anda dalam repositori pribadi. Untuk informasi selengkapnya, lihat the section called “Mengakses gambar Docker dari repositori pribadi”.

Nama tampilan pada konsol AWS IoT tersebut: Kredensial untuk repositori privat

Diperlukan:`false`. Parameter ini diperlukan untuk mengakses citra Docker yang dipertahankan dalam repositori privat.

Jenis: `array` dari `string`

Pola yang valid: `[( ?,? ?"(arn:(aws(-[a-z]+)):secretsmanager:[a-z0-9-]+:[0-9]{12}:secret:([a-zA-Z0-9\]+/)[a-zA-Z0-9/_+=,.@-]+-[a-zA-Z0-9]+)")]`

`DockerContainerStatusLogFrequency`

Frekuensi (dalam detik) saat konektor mencatat informasi status tentang kontainer Docker yang berjalan pada core. Default-nya adalah 300 detik (5 menit).

Nama tampilan pada konsol AWS IoT tersebut: Frekuensi pencatatan

Wajib: `false`

Jenis: `string`

Pola yang valid: `^[1-9]{1}[0-9]{0,3}$`

Buat Contoh Konektor (AWS CLI)

Perintah CLI berikut membuat `ConnectorDefinition` dengan versi awal yang berisi konektor deployment aplikasi Docker Greengrass.

```
aws greengrass create-connector-definition --name MyGreengrassConnectors --initial-
version '{
    "Connectors": [
        {
            "Id": "MyDockerAppplicationDeploymentConnector",
            "ConnectorArn": "arn:aws:greengrass:region::/connectors/
DockerApplicationDeployment/versions/5",
            "Parameters": {
                "DockerComposeFileS3Bucket": "myS3Bucket",
                "DockerComposeFileS3Key": "production-docker-compose.yml",
                "DockerComposeFileS3Version": "123",
                "DockerComposeFileDestinationPath": "/home/username/myCompose",
                "DockerUserId": "1000",
                "AWSSecretsArnList": "[\"arn:aws:secretsmanager:region:account-
id:secret:greengrass-secret1-hash\",\"arn:aws:secretsmanager:region:account-
id:secret:greengrass-secret2-hash\"]",
                "DockerContainerStatusLogFrequency": "30",
                "ForceDeploy": "True",
                "DockerPullBeforeUp": "True"
            }
```

```
        }
    ]
}'
```

> **Note**
>
> Fungsi Lambda dalam konektor mempunyai siklus hidup yang berumur panjang ini.

Data input

Konektor ini tidak memerlukan atau menerima data input.

Data output

Konektor ini menerbitkan status `docker-compose up` perintah sebagai data output.

Filter topik dalam langganan

`dockerapplicationdeploymentconnector/message/status`

Contoh keluaran: Sukses

```
{
  "status":"success",
  "GreengrassDockerApplicationDeploymentStatus":"Successfully triggered docker-
compose up",
  "S3Bucket":"myS3Bucket",
  "ComposeFileName":"production-docker-compose.yml",
  "ComposeFileVersion":"123"
}
```

Contoh keluaran: Kegagalan

```
{
  "status":"fail",
  "error_message":"description of error",
  "error":"InvalidParameter"
}
```

Jenis kesalahan dapat berupa `InvalidParameter` atau `InternalError`.

Mengatur pengguna Docker pada core AWS IoT Greengrass ini

Konektor deployment aplikasi Docker Greengrass berjalan sebagai pengguna yang Anda tentukan untuk `DockerUserId` parameter. Jika Anda tidak menentukan nilai, konektor berjalan sebagai `ggc_user`, yang merupakan default identitas akses Greengrass.

Untuk mengizinkan konektor berinteraksi dengan Docker daemon, pengguna Docker harus milik grup Linux `docker` pada core. Pengguna Docker juga harus memiliki izin menulis ke direktori `DockerComposeFileDestinationPath` ini. Di sinilah konektor menyimpan file lokal `docker-compose.yml` dan kredensial Docker Anda.

> **Note**
>
> - Kami merekomendasikan Anda membuat pengguna Linux daripada menggunakan default `ggc_user`. Jika tidak, fungsi Lambda dalam grup Greengrass dapat mengakses file Compose dan kredensial Docker.
> - Kami merekomendasikan bahwa Anda menghindari berjalan sebagai root kecuali benar-benar diperlukan. Jika Anda menentukan pengguna root, Anda harus mengizinkan fungsi Lambda untuk menjalankan sebagai root pada AWS IoT Greengrass core. Untuk informasi selengkapnya, lihat the section called “Menjalankan fungsi Lambda sebagai root”.

1. Buat pengguna. Anda dapat menjalankan perintah `useradd` dan termasuk opsional pada pilihan `-u` untuk menetapkan UID. Sebagai contoh:

   ```
   sudo useradd -u 1234 user-name
   ```

2. Tambahkan pengguna ke `docker` grup pada core. Sebagai contoh:

   ```
   sudo usermod -aG docker user-name
   ```

   Untuk informasi lebih lanjut, termasuk cara membuat `docker` grup, lihat Mengelola Docker sebagai pengguna non-root dalam dokumentasi Docker.

3. Berikan izin pengguna untuk menulis ke direktori yang ditentukan untuk `DockerComposeFileDestinationPath` parameter. Sebagai contoh:

   a. Untuk mengatur pengguna sebagai pemilik direktori. Contoh ini menggunakan UID dari langkah 1.

```
chown 1234 docker-compose-file-destination-path
```

b. Untuk memberikan izin baca dan tulis kepada pemilik.

```
chmod 700 docker-compose-file-destination-path
```

Untuk informasi lebih lanjut, lihat Cara mengelola File dan Folder Izin dalam Linux dalam dokumentasi Linux Foundation.

c. Jika Anda tidak menetapkan UID ketika membuat pengguna, atau jika Anda menggunakan pengguna yang sudah ada, jalankan `id` perintah untuk mencari UID.

```
id -u user-name
```

Anda menggunakan UID untuk mengonfigurasi `DockerUserId` parameter untuk konektor.

Informasi penggunaan

Ketika Anda menggunakan konektor deployment aplikasi Docker Greengrass, Anda harus mengetahui informasi penggunaan spesifik implementasi berikut.

- Awalan tetap untuk nama proyek. Konektor prepends `greengrassdockerapplicationdeployment` awalan untuk nama-nama kontainer Docker yang memulai. Konektor menggunakan awalan ini sebagai nama proyek dalam perintah `docker-compose` yang berjalan.
- Perilaku logging. Konektor menulis informasi status dan informasi penyelesaian masalah ke file log. Anda dapat mengonfigurasi AWS IoT Greengrass untuk mengirim CloudWatch log ke Log dan menulis log secara lokal. Untuk informasi selengkapnya, lihat the section called “Mencatat”. Ini adalah jalur ke log lokal untuk konektor:

  ```
  /greengrass-root/ggc/var/log/user/region/aws/DockerApplicationDeployment.log
  ```

  Anda harus memiliki izin root untuk mengakses log lokal.
- Memperbarui gambar Docker. Docker menyimpan gambar pada perangkat core. Jika Anda memperbarui gambar Docker dan ingin menerapkankan perubahan ke perangkat core, pastikan untuk mengubah tag untuk gambar dalam file Compose. Perubahan membawa efek setelah grup Greengrass diterapkan.

- Batas waktu 10 menit untuk operasi pembersihan. Ketika Greengrass daemon berhenti selama restart, `docker-compose down` perintah dimulai. Semua kontainer Docker memiliki maksimum 10 menit setelah `docker-compose down` dimulai untuk melakukan operasi pembersihan apa pun. Jika pembersihan tidak selesai dalam 10 menit, Anda harus membersihkan kontainer yang tersisa secara manual. Untuk informasi lebih lanjut, lihat docker rm dalam dokumentasi Docker CLI.
- Menjalankan perintah Docker. Untuk menyelesaikan masalah, Anda dapat menjalankan perintah Docker dalam jendela terminal pada perangkat core. Sebagai contoh, jalankan perintah berikut untuk melihat Docker kontainer yang dimulai oleh konektor:

  ```
  docker ps --filter name="greengrassdockerapplicationdeployment"
  ```

- ID sumber daya yang dicadangkan. Konektor menggunakan `DOCKER_DEPLOYER_SECRET_RESOURCE_RESERVED_ID_`*`index`* ID untuk sumber daya Greengrass me buat dalam grup Greengrass. ID sumber daya harus unik dalam grup, jadi jangan tetapkan ID sumber daya yang mungkin bertentangan dengan ID sumber daya cadangan.
- Mode offline. Ketika Anda mengatur `DockerOfflineMode` parameter konfigurasi untuk `True`, maka konektor Docker dapat beroperasi dalam mode offline. Hal ini dapat terjadi ketika deployment grup Greengrass restart sementara perangkat core offline, dan konektor tidak dapat membuat koneksi ke Amazon S3 atau Amazon ECR untuk mengambil file Docker Compose.

  Dengan mode offline diaktifkan, konektor akan mencoba mengunduh file Compose, dan menjalankan `docker login` perintah seperti itu akan untuk restart normal. Jika upaya ini gagal, maka konektor mencari dipertahankan secara lokal file Compose dalam folder yang ditentukan menggunakan `DockerComposeFileDestinationPath` parameter. Jika ada file Compose lokal, maka konektor mengikuti urutan normal `docker-compose` perintah dan menarik dari gambar lokal. Jika menulis file atau gambar lokal tidak ada, maka konektor gagal. Perilaku parameter `ForceDeploy` dan `StopContainersOnNewDeployment` tetap sama dalam mode offline.

Berkomunikasi dengan kontainer Docker

AWS IoT Greengrass mendukung saluran komunikasi berikut antara komponen Greengrass dan kontainer Docker:

- Fungsi Greengrass Lambda dapat menggunakan API REST untuk berkomunikasi dengan proses dalam kontainer Docker. Anda dapat mengatur server dalam kontainer Docker yang membuka port. Fungsi Lambda dapat berkomunikasi dengan kontainer dalam port ini.

- Proses dalam kontainer Docker dapat bertukar pesan MQTT melalui broker pesan Greengrass lokal. Anda dapat mengatur wadah Docker sebagai perangkat klien di grup Greengrass dan kemudian membuat langganan untuk memungkinkan penampung berkomunikasi dengan fungsi Greengrass Lambda, perangkat klien, dan konektor lain dalam grup, atau dengan dan layanan bayangan lokal. AWS IoT Untuk informasi selengkapnya, lihat the section called “Mengonfigurasi komunikasi MQTT dengan kontainer Docker”.
- Fungsi Greengrass Lambda dapat memperbarui file bersama untuk menyampaikan informasi ke kontainer Docker. Anda dapat menggunakan file menulis untuk mengikat mount jalur file bersama untuk kontainer Docker.

Mengonfigurasi komunikasi MQTT dengan kontainer Docker

Anda dapat mengonfigurasi wadah Docker sebagai perangkat klien dan menambahkannya ke grup Greengrass. Kemudian, Anda dapat membuat langganan yang mengizinkan komunikasi MQTT antara kontainer Docker dan komponen Greengrass atau AWS IoT. Dalam prosedur berikut, Anda membuat langganan yang mengizinkan perangkat kontainer Docker untuk menerima bayangan pembaruan pesan dari layanan bayangan lokal. Anda dapat mengikuti pola ini untuk membuat langganan lainnya.

> **Note**
>
> Prosedur ini mengasumsikan bahwa Anda telah membuat grup Greengrass dan core Greengrass (v1.10 atau yang lebih baru). Untuk informasi tentang pembuatan grup dan core Greengrass, lihat Memulai dengan AWS IoT Greengrass.

Untuk mengonfigurasi wadah Docker sebagai perangkat klien dan menambahkannya ke grup Greengrass

1. Buat folder pada perangkat inti untuk menyimpan sertifikat dan kunci yang digunakan untuk mengautentikasi perangkat Greengrass.

   Jalur file harus dipasang pada kontainer Docker Anda ingin memulai. Potongan berikut menunjukkan cara me-mount jalur file dalam file Compose Anda. Dalam contoh ini, *`path-to-device-certs`*mewakili folder yang Anda buat di langkah ini.

   ```
   version: '3.3'
   services:
   ```

```
  myService:
    image: user-name/repo:image-tag
    volumes:
      - /path-to-device-certs/:/path-accessible-in-container
```

2. Di panel navigasi AWS IoT konsol, di bawah Kelola, perluas perangkat Greengrass, lalu pilih Grup (V1).
3. Pilih grup target.
4. Pada halaman konfigurasi grup, pilih Perangkat klien, lalu pilih Associate.
5. Di Mengaitkan perangkat klien dengan modal grup ini, pilih Buat AWS IoT hal baru.

   Halaman Create things terbuka di tab baru.
6. Pada halaman Create things, pilih Create single thing, lalu pilih Next.
7. Pada halaman Tentukan properti benda, masukkan nama untuk perangkat, lalu pilih Berikutnya.
8. Pada halaman Konfigurasi sertifikat perangkat, pilih Berikutnya.
9. Pada halaman Lampirkan kebijakan ke sertifikat, lakukan salah satu hal berikut:
   - Pilih kebijakan yang ada yang memberikan izin yang diperlukan perangkat klien, lalu pilih Buat sesuatu.

     Modal terbuka di mana Anda dapat mengunduh sertifikat dan kunci yang digunakan perangkat untuk terhubung ke AWS Cloud dan inti.
   - Buat dan lampirkan kebijakan baru yang memberikan izin perangkat klien. Lakukan hal-hal berikut:

     a. Pilih Buat kebijakan.

        Halaman Buat kebijakan terbuka di tab baru.

     b. Pada halaman Buat kebijakan, lakukan hal berikut:

        i. Untuk nama Kebijakan, masukkan nama yang menjelaskan kebijakan, seperti**`GreengrassV1ClientDevicePolicy`**.

        ii. Pada tab Pernyataan kebijakan, di bawah Dokumen kebijakan, pilih JSON.

        iii. Masukkan dokumen kebijakan berikut. Kebijakan ini memungkinkan perangkat klien untuk menemukan inti Greengrass dan berkomunikasi pada semua topik MQTT. Untuk informasi tentang cara membatasi akses kebijakan ini, lihatAutentikasi dan otorisasi perangkat untuk AWS IoT Greengrass.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Subscribe",
        "iot:Connect",
        "iot:Receive"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "greengrass:*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

iv. Pilih Buat untuk membuat kebijakan.

c. Kembali ke tab browser dengan halaman Lampirkan kebijakan ke sertifikat terbuka. Lakukan hal-hal berikut:

i. Dalam daftar Kebijakan, pilih kebijakan yang Anda buat, seperti GreengrassV1ClientDevicePolicy.

Jika Anda tidak melihat kebijakan, pilih tombol segarkan.

ii. Pilih Buat sesuatu.

Modal terbuka di mana Anda dapat mengunduh sertifikat dan kunci yang digunakan perangkat untuk terhubung ke AWS Cloud dan inti.

10. Dalam modal Unduh sertifikat dan kunci, unduh sertifikat perangkat.

> **Important**
>
> Sebelum Anda memilih Selesai, unduh sumber daya keamanan.

Lakukan hal-hal berikut:

a. Untuk sertifikat Perangkat, pilih Unduh untuk mengunduh sertifikat perangkat.

b. Untuk file kunci Publik, pilih Unduh untuk mengunduh kunci publik untuk sertifikat.

c. Untuk file kunci pribadi, pilih Unduh untuk mengunduh file kunci pribadi untuk sertifikat.

d. Review Autentikasi Server dalam AWS IoT Panduan Developer dan memilih root sertifikat CA yang sesuai. Kami merekomendasikan Anda menggunakan Amazon Trust Services (ATS) titik akhir dan ATS root sertifikat CA. Di bawah sertifikat Root CA, pilih Unduh untuk sertifikat CA root.

e. Pilih Selesai.

Catat ID sertifikat yang umum dalam nama file untuk sertifikat dan kunci perangkat. Anda membutuhkannya nanti.

11. Salin sertifikat dan kunci ke folder yang Anda buat di langkah 1.

Selanjutnya, buat langganan dalam grup. Untuk contoh ini, Anda buat langganan mengizinkan perangkat kontainer Docker untuk menerima pesan MQTT dari layanan bayangan lokal.

> **Note**
>
> Ukuran maksimum dokumen bayangan adalah 8 KB. Untuk informasi lebih lanjut, lihat AWS IoT kuota dalam AWS IoT Panduan Developer.

Untuk membuat langganan yang mengizinkan perangkat kontainer Docker menerima pesan MQTT dari layanan bayangan lokal

1. Pada halaman konfigurasi grup, pilih tab Langganan, lalu pilih Tambah Langganan.
2. Pada halaman Pilih sumber dan target Anda ini, konfigurasikan sumber dan target, sebagai berikut:

a. Untuk Pilihan sumber, pilih Layanan, dan kemudian pilih Layanan Bayangan Lokal.

b. Untuk Pilih target, pilih Perangkat, kemudian pilih perangkat Anda.

c. Pilih Berikutnya.

d. Pada halaman Filter data Anda dengan topik ini, untuk Filter topik, pilih **$aws/things/*MyDockerDevice*/shadow/update/accepted**, kemudian pilih Selanjutnya. Ganti *MyDockerDevice*dengan nama perangkat yang Anda buat sebelumnya.

e. Pilih Selesai.

Sertakan potongan kode berikut dalam gambar Docker yang Anda referensi dalam file Compose. Ini adalah kode perangkat Greengrass. Juga, tambahkan kode dalam kontainer Docker Anda memulai perangkat Greengrass dalam kontainer. Ini dapat berjalan sebagai proses terpisah dalam gambar atau dalam thread terpisah.

```
import os
import sys
import time
import uuid

from AWSIoTPythonSDK.core.greengrass.discovery.providers import DiscoveryInfoProvider
from AWSIoTPythonSDK.exception.AWSIoTExceptions import DiscoveryInvalidRequestException
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient

# Replace thingName with the name you registered for the Docker device.
thingName = "MyDockerDevice"
clientId = thingName

# Replace host with the IoT endpoint for your &AWS-account;.
host = "myPrefix.iot.region.amazonaws.com"

# Replace topic with the topic where the Docker container subscribes.
topic = "$aws/things/MyDockerDevice/shadow/update/accepted"

# Replace these paths based on the download location of the certificates for the Docker
 container.
rootCAPath = "/path-accessible-in-container/AmazonRootCA1.pem"
certificatePath = "/path-accessible-in-container/certId-certificate.pem.crt"
privateKeyPath = "/path-accessible-in-container/certId-private.pem.key"
```

```
# Discover Greengrass cores.
discoveryInfoProvider = DiscoveryInfoProvider()
discoveryInfoProvider.configureEndpoint(host)
discoveryInfoProvider.configureCredentials(rootCAPath, certificatePath, privateKeyPath)
discoveryInfoProvider.configureTimeout(10)  # 10 seconds.

GROUP_CA_PATH = "./groupCA/"
MQTT_QOS = 1

discovered = False
groupCA = None
coreInfo = None

try:
    # Get discovery info from AWS IoT.
    discoveryInfo = discoveryInfoProvider.discover(thingName)
    caList = discoveryInfo.getAllCas()
    coreList = discoveryInfo.getAllCores()

    # Use first discovery result.
    groupId, ca = caList[0]
    coreInfo = coreList[0]

    # Save the group CA to a local file.
    groupCA = GROUP_CA_PATH + groupId + "_CA_" + str(uuid.uuid4()) + ".crt"
    if not os.path.exists(GROUP_CA_PATH):
        os.makedirs(GROUP_CA_PATH)
    groupCAFile = open(groupCA, "w")
    groupCAFile.write(ca)
    groupCAFile.close()
    discovered = True
except DiscoveryInvalidRequestException as e:
    print("Invalid discovery request detected!")
    print("Type: %s" % str(type(e)))
    print("Error message: %s" % str(e))
    print("Stopping...")
except BaseException as e:
    print("Error in discovery!")
    print("Type: %s" % str(type(e)))
    print("Error message: %s" % str(e))
    print("Stopping...")

myAWSIoTMQTTClient = AWSIoTMQTTClient(clientId)
myAWSIoTMQTTClient.configureCredentials(groupCA, privateKeyPath, certificatePath)
```

```
# Try to connect to the Greengrass core.
connected = False
for connectivityInfo in coreInfo.connectivityInfoList:
    currentHost = connectivityInfo.host
    currentPort = connectivityInfo.port
    myAWSIoTMQTTClient.configureEndpoint(currentHost, currentPort)
    try:
        myAWSIoTMQTTClient.connect()
        connected = True
    except BaseException as e:
        print("Error in connect!")
        print("Type: %s" % str(type(e)))
        print("Error message: %s" % str(e))
    if connected:
        break

if not connected:
    print("Cannot connect to core %s. Exiting..." % coreInfo.coreThingArn)
    sys.exit(-2)

# Handle the MQTT message received from GGShadowService.
def customCallback(client, userdata, message):
    print("Received an MQTT message")
    print(message)

# Subscribe to the MQTT topic.
myAWSIoTMQTTClient.subscribe(topic, MQTT_QOS, customCallback)

# Keep the process alive to listen for messages.
while True:
    time.sleep(1)
```

Catatan keamanan

Ketika Anda menggunakan konektor deployment aplikasi Docker Greengrass, perhatikan pertimbangan keamanan berikut.

Penyimpanan lokal file Docker Compose

Konektor menyimpan salinan file Compose Anda dalam direktori yang ditentukan untuk `DockerComposeFileDestinationPath` parameter.

Ini adalah tanggung jawab Anda untuk mengamankan direktori ini. Anda harus menggunakan izin sistem file untuk membatasi akses ke direktori.

Penyimpanan lokal kredenal Docker

Jika citra Docker Anda dipertahankan dalam repositori privat, konektor menyimpan kredensial Docker Anda dalam direktori yang ditentukan untuk `DockerComposeFileDestinationPath` Parameter.

Ini adalah tanggung jawab Anda untuk mengamankan kredensial ini. Sebagai contoh, Anda harus menggunakan pembantu-kredensial pada perangkat core ketika Anda menginstal Docker Engine.

Instal Docker Engine dari sumber tepercaya

Anda bertanggung jawab untuk menginstal Docker Engine dari sumber terpercaya. Konektor ini menggunakan Docker daemon pada perangkat core untuk mengakses aset Docker Anda dan mengelola kontainer Docker.

Lingkup izin peran grup Greengrass

Izin yang Anda tambahkan dalam peran grup Greengrass dapat diasumsikan oleh semua fungsi Lambda dan konektor dalam grup Greengrass. Konektor ini memerlukan akses ke file Docker Anda yang dipertahankan dalam bucket S3. Hal ini juga membutuhkan akses ke token otorisasi Amazon ECR Anda jika gambar Docker Anda dipertahankan dalam repositori pribadi dalam Amazon ECR.

Lisensi

Konektor deployment aplikasi Docker Greengrass termasuk perangkat lunak/lisensi pihak ketiga berikut:

- AWS SDK for Python (Boto3)/Lisensi 2.0 Apache
- botocore/Apache License 2.0
- dateutil/PSF License
- docutils/BSD License, Lisensi Publik Umum (GPL) GNU, Lisensi Dasar Perangkat Lunak Python, Domain Publik
- jmespath/MIT License

- s3transfer/Apache License 2.0
- urllib3/Lisensi MIT

Konektor ini dirilis di bawah Perjanjian Lisensi Perangkat lunak core Greengrass.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi konektor.

Versi	Perubahan
7	Ditambahkan `DockerOfflineMode` untuk menggunakan file Docker Compose yang ada ketika AWS IoT Greengrass mulai offline. Diimplementasikan coba lagi untuk `docker login` perintah. Mendukung untuk UID 32-bit.
6	Ditambahkan `StopContainersOnNewDeployment` untuk menimpa kontainer membersihkan ketika deployment baru dibuat atau GC berhenti. Mekanisme shutdown dan start up yang lebih aman. YAML validation bug fix.
5	Gambar ditarik sebelum menjalankan `docker-compose down` .
4	Menambahkan pull-before-up perilaku untuk memperbarui gambar Docker.
3	Diperbaiki masalah dengan menemukan variabel lingkungan.
2	Ditambahkan `ForceDeploy` parameter.
1	Pelepasan awal.

Sebuah grup Greengrass dapat berisi hanya satu versi konektor pada suatu waktu. Untuk informasi lebih lanjut tentang pembaruan versi konektor, lihat the section called “Versi upgrade konektor”.

Lihat juga

- Integrasikan dengan layanan dan protokol menggunakan konektor
- the section called “Memulai dengan konektor (konsol)”
- the section called “Memulai dengan konektor (CLI)”

Konektor IoT Analytics

> **Warning**
>
> Konektor ini telah pindah ke fase masa pakai yang diperpanjang, danAWS IoT Greengrass tidak akan merilis pembaruan yang menyediakan fitur, penyempurnaan pada fitur yang ada, tambalan keamanan, atau perbaikan bug. Untuk informasi selengkapnya, lihat AWS IoT Greengrass Version 1kebijakan pemeliharaan.

Konektor IoT Analytics mengirimkan data perangkat lokal ke AWS IoT Analytics. Anda dapat menggunakan konektor ini sebagai hub pusat untuk mengumpulkan data dari sensor pada perangkat core Greengrass dan dari perangkat klien yang terhubung. Konektor mengirimkan data ke AWS IoT Analytics saluran dalam arus Akun AWS dan Wilayah. Hal ini dapat mengirim data ke saluran tujuan default dan saluran yang secara dinamis ditentukan.

> **Note**
>
> AWS IoT Analytics adalah layanan yang dikelola sepenuhnya yang mengizinkan Anda untuk mengumpulkan, menyimpan, memproses, dan menanyakan data IoT. Dalam AWS IoT Analytics, data dapat dianalisis dan diproses lebih lanjut. Sebagai contoh, ini dapat digunakan untuk melatih model ML untuk memantau kondisi mesin atau untuk menguji strategi pemodelan baru. Untuk informasi lebih lanjut, lihat Apa AWS IoT Analytics? dalam AWS IoT Analytics Panduan Pengguna.

Konektor menerima data yang diformat dan tidak diformat pada input topik MQTT. Mendukung dua topik yang telah ditetapkan dalam mana saluran tujuan ditentukan inline. Hal ini juga dapat menerima pesan pada topik yang ditetapkan pelanggan yang dikonfigurasi dalam langganan. Hal ini dapat digunakan untuk merutekan pesan dari perangkat klien yang menerbitkan ke topik tetap atau menangani data yang tidak terstruktur atau tergantung tumpukan dari perangkat yang dibatasi sumber daya.

Konektor ini menggunakan `BatchPutMessage`API untuk mengirim data (sebagai JSON atau base64 encoded string) ke saluran tujuan. Konektor dapat memproses data mentah ke dalam format yang sesuai dengan persyaratan API. Konektor buffer pesan input dalam per-channel antrian dan asynchronously memproses batch. Ini menyediakan parameter yang mengizinkan Anda untuk mengontrol antrian dan perilaku batching dan membatasi konsumsi memori. Sebagai contoh, Anda

dapat mengonfigurasi ukuran antrian maksimum, interval batch, ukuran memori, dan jumlah saluran aktif.

Konektor ini memiliki versi berikut.

Versi	ARN
4	`arn:aws:greengrass:region::/connectors/IoTAnalytics/versions/4`
3	`arn:aws:greengrass:region::/connectors/IoTAnalytics/versions/3`
2	`arn:aws:greengrass:region::/connectors/IoTAnalytics/versions/2`
1	`arn:aws:greengrass:region::/connectors/IoTAnalytics/versions/1`

Untuk informasi tentang perubahan versi, lihat Changelog.

Persyaratan

Konektor ini memiliki persyaratan sebagai berikut:

Version 3 - 4

- AWS IoT Greengrass perangkat lunak Core v1.9.3 atau yang lebih baru.
- Python versi 3.7 atau 3.8 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.

> **Note**
>
> Untuk menggunakan Python 3.8, jalankan perintah berikut untuk membuat link simbolik dari folder instalasi default Python 3.7 ke binari Python 3.8 yang diinstal.

```
sudo ln -s path-to-python-3.8/python3.8 /usr/bin/python3.7
```

Ini mengonfigurasi perangkat Anda untuk memenuhi persyaratan Python untuk AWS IoT Greengrass.

- Konektor ini hanya dapat digunakan dalam Wilayah Amazon Web Services di mana kedua AWS IoT Greengrass dan AWS IoT Analytics didukung.
- Semua terkait entitas AWS IoT Analytics dan alur kerja dibuat dan dikonfigurasi. Entitas termasuk saluran, alur, penyimpanan data, dan set data. Untuk informasi lebih lanjut, lihat AWS CLI atau prosedur konsol dalam AWS IoT Analytics Panduan Pengguna.

> **Note**
>
> Tujuan AWS IoT Analytics saluran harus menggunakan akun yang sama dan ada dalam Wilayah AWS sebagai konektor ini.

- Pada Peran grup Greengrass dikonfigurasi untuk mengizinkan `iotanalytics:BatchPutMessage` tindakan dalam saluran tujuan, seperti yang ditunjukkan dalam kebijakan IAM berikut. Saluran harus dalam arus Akun AWS dan Wilayah.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1528133056761",
            "Action": [
                "iotanalytics:BatchPutMessage"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iotanalytics:region:account-id:channel/channel_1_name",
                "arn:aws:iotanalytics:region:account-id:channel/channel_2_name"
            ]
        }
    ]
}
```

Untuk persyaratan peran grup, Anda harus mengonfigurasi peran untuk memberikan izin yang diperlukan dan memastikan peran telah ditambahkan ke grup. Untuk informasi lebih lanjut, lihat the section called “Mengelola peran grup (konsol)” atau the section called “Kelola peran grup (CLI)”.

Versions 1 - 2

- AWS IoT Greengrass perangkat lunak Core v1.7 atau yang lebih baru.
- Python versi 2.7 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.
- Konektor ini hanya dapat digunakan dalam Wilayah Amazon Web Services di mana kedua AWS IoT Greengrass dan AWS IoT Analytics didukung.
- Semua terkait entitas AWS IoT Analytics dan alur kerja dibuat dan dikonfigurasi. Entitas termasuk saluran, alur, penyimpanan data, dan set data. Untuk informasi lebih lanjut, lihat AWS CLI atau prosedur konsol dalam AWS IoT Analytics Panduan Pengguna.

  > **Note**
  >
  > Tujuan AWS IoT Analytics saluran harus menggunakan akun yang sama dan ada dalam Wilayah AWS sebagai konektor ini.

- Pada Peran grup Greengrass dikonfigurasi untuk mengizinkan `iotanalytics:BatchPutMessage` tindakan dalam saluran tujuan, seperti yang ditunjukkan dalam kebijakan IAM berikut. Saluran harus dalam arus Akun AWS dan Wilayah.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1528133056761",
            "Action": [
                "iotanalytics:BatchPutMessage"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iotanalytics:region:account-id:channel/channel_1_name",
                "arn:aws:iotanalytics:region:account-id:channel/channel_2_name"
            ]
        }
```

```
    ]
}
```

Untuk persyaratan peran grup, Anda harus mengonfigurasi peran untuk memberikan izin yang diperlukan dan memastikan peran telah ditambahkan ke grup. Untuk informasi lebih lanjut, lihat the section called “Mengelola peran grup (konsol)” atau the section called “Kelola peran grup (CLI)”.

Parameter

`MemorySize`

Jumlah memori (dalam KB) untuk mengalokasikan ke konektor ini.

Nama tampilan pada konsol AWS IoT tersebut: Ukuran memori

Wajib: `true`

Jenis: `string`

Pola yang valid: `^[0-9]+$`

`PublishRegion`

Pada Wilayah AWS dalam mana saluran AWS IoT Analytics Anda dibuat. Gunakan Wilayah yang sama dengan konektornya.

> **Note**
>
> Ini juga harus sesuai dengan Wilayah untuk saluran yang ditentukan dalam peran grup.

Nama tampilan pada konsol AWS IoT tersebut: Publikasikan wilayah

Wajib: `false`

Jenis: `string`

Pola yang valid: `^$|([a-z]{2}-[a-z]+-\\d{1})`

`PublishInterval`

Interval (dalam detik) untuk menerbitkan batch data yang diterima ke AWS IoT Analytics.

Nama tampilan pada konsol AWS IoT tersebut: Publikasikan Interval

Wajib: `false`

Jenis: `string`

Nilai default: `1`

Pola yang valid: `$|^[0-9]+$`

`IotAnalyticsMaxActiveChannels`

Jumlah maksimum AWS IoT Analytics saluran yang konektor secara aktif menonton untuk. Ini harus lebih besar dari 0, dan setidaknya sama dengan jumlah saluran yang Anda harapkan konektor untuk menerbitkankan ke pada waktu tertentu.

Anda dapat menggunakan parameter ini untuk membatasi konsumsi memori dengan membatasi jumlah antrian yang konektor dapat mengelola pada waktu tertentu. Antrian dihapus ketika semua pesan antrian dikirim.

Nama tampilan pada konsol AWS IoT tersebut: Jumlah maksimum saluran aktif

Wajib: `false`

Jenis: `string`

Nilai default: `50`

Pola yang valid: `^$|^[1-9][0-9]*$`

`IotAnalyticsQueueDropBehavior`

Perilaku untuk batal pesan dari antrian saluran ketika antrian penuh.

Nama tampilan pada konsol AWS IoT tersebut: Perilaku batal antrean

Wajib: `false`

Jenis: `string`

Nilai yang valid: `DROP_NEWEST` or `DROP_OLDEST`

Nilai default: `DROP_NEWEST`

Pola yang valid: `^DROP_NEWEST$|^DROP_OLDEST$`

`IotAnalyticsQueueSizePerChannel`

Jumlah maksimum pesan untuk mempertahankan dalam memori (per saluran) sebelum pesan dikirim atau dibatalkan. Ini harus lebih besar dari 0.

Nama tampilan pada konsol AWS IoT tersebut: Ukuran antrean maksimum per saluran

Wajib: `false`

Jenis: `string`

Nilai default: `2048`

Pola yang valid: `^$|^[1-9][0-9]*$`

`IotAnalyticsBatchSizePerChannel`

Jumlah maksimum pesan yang akan dikirim ke AWS IoT Analytics dalam satu permintaan batch. Ini harus lebih besar dari 0.

Nama tampilan pada konsol AWS IoT tersebut: Jumlah maksimum pesan untuk batch per saluran

Wajib: `false`

Jenis: `string`

Nilai default: `5`

Pola yang valid: `^$|^[1-9][0-9]*$`

`IotAnalyticsDefaultChannelName`

Nama AWS IoT Analytics saluran yang menggunakan konektor ini untuk pesan yang dikirim ke topik input yang ditetapkan pelanggan.

Nama tampilan pada konsol AWS IoT tersebut: Nama saluran default

Wajib: `false`

Jenis: `string`

Pola yang valid: `^[a-zA-Z0-9_]$`

`IsolationMode`

Mode kontainerisasi untuk konektor ini. Default-nya adalah `GreengrassContainer`, yang berarti bahwa konektor berjalan dalam lingkungan waktu aktif terisolasi dalam kontainer AWS IoT Greengrass ini.

> **Note**
>
> Pengaturan kontainerisasi default untuk grup tidak berlaku untuk konektor.

Nama tampilan pada konsol AWS IoT tersebut: Mode isolasi kontainer

Wajib: `false`

Jenis: `string`

Nilai yang valid: `GreengrassContainer` or `NoContainer`

Pola yang valid: `^NoContainer$|^GreengrassContainer$`

Buat Contoh Konektor (AWS CLI)

Perintah CLI berikut membuat `ConnectorDefinition` dengan versi awal yang berisi konektor IoT Analytics.

```
aws greengrass create-connector-definition --name MyGreengrassConnectors --initial-
version '{
    "Connectors": [
        {
            "Id": "MyIoTAnalyticsApplication",
            "ConnectorArn": "arn:aws:greengrass:region::/connectors/IoTAnalytics/
versions/3",
            "Parameters": {
                "MemorySize": "65535",
                "PublishRegion": "us-west-1",
                "PublishInterval": "2",
                "IotAnalyticsMaxActiveChannels": "25",
                "IotAnalyticsQueueDropBehavior": "DROP_OLDEST",
                "IotAnalyticsQueueSizePerChannel": "1028",
                "IotAnalyticsBatchSizePerChannel": "5",
                "IotAnalyticsDefaultChannelName": "my_channel"
```

```
            }
        }
    ]
}'
```

> **Note**
>
> Fungsi Lambda dalam konektor mempunyai siklus hidup yang berumur panjang ini.

Di konsol AWS IoT Greengrass tersebut, Anda dapat menambahkan konektor dari halaman grup Konektor ini. Untuk informasi lebih lanjut, lihat the section called “Memulai dengan konektor (konsol)”.

Data input

Konektor ini menerima data pada topik MQTT yang telah ditetapkan dan ditetapkan pelanggan. Penerbit dapat berupa perangkat klien Lambda atau konektor lainnya.

Topik yang telah ditetapkan

Konektor mendukung berikut dua terstruktur MQTT topik yang mengizinkan penerbit untuk menentukan nama saluran inline.

- Sebuah pesan yang telah diformat pada `iotanalytics/channels/+/messages/put` topik. Data IoT dalam pesan input ini harus diformat sebagai JSON atau base64-encoded string.
- Pesan yang belum diformat pada `iotanalytics/channels/+/messages/binary/put` topik. Pesan input yang diterima pada topik ini diperlakukan sebagai data biner dan dapat berisi jenis data.

  Untuk menerbitkankan ke topik yang telah ditetapkan, ganti + wildcard dengan nama saluran. Misalnya:

  ```
  iotanalytics/channels/my_channel/messages/put
  ```

Topik yang ditetapkan pelanggan

Konektor mendukung # topik sintaks, yang mengizinkan untuk menerima pesan input pada setiap topik MQTT yang Anda konfigurasi dalam langganan. Kami menyarankan Anda menentukan jalur topik alih-alih hanya menggunakan # wildcard dalam langganan Anda. Pesan ini dikirim ke saluran default yang Anda tentukan untuk konektor.

Pesan input pada topik yang ditetapkan pelanggan diperlakukan sebagai data biner. Mereka dapat menggunakan format pesan dan dapat berisi jenis data. Anda dapat menggunakan topik yang ditetapkan pelanggan untuk merutekan pesan dari perangkat yang diterbitkankan ke topik tetap. Anda juga dapat menggunakannya untuk menerima data input dari perangkat klien yang tidak dapat memproses data ke dalam pesan yang diformat untuk dikirim ke konektor.

Untuk informasi lebih lanjut tentang langganan dan topik MQTT, lihat the section called “Input dan output”.

Peran grup harus mengizinkan tindakan `iotanalytics:BatchPutMessage` pada semua bucket tujuan. Untuk informasi selengkapnya, lihat the section called “Persyaratan”.

Filter topik: `iotanalytics/channels/+/messages/put`

Gunakan topik ini untuk mengirim pesan yang diformat ke konektor dan secara dinamis menentukan saluran tujuan. Topik ini juga mengizinkan Anda untuk menentukan ID yang dikembalikan dalam output respon. Konektor memverifikasi bahwa ID unik untuk setiap pesan dalam outbound `BatchPutMessage` meminta agar mengirimkannya ke AWS IoT Analytics. Pesan yang memiliki ID duplikat dibatalkan.

Data input yang dikirim ke topik ini harus menggunakan format pesan berikut.

Properti pesan

`request`

Data untuk mengirim ke saluran tertentu.

Wajib: `true`

Jenis: `object` yang mencakup properti berikut:

`message`

Perangkat atau sensor data sebagai JSON atau base64-encoded string.

Wajib: `true`

Jenis: `string`

`id`

ID arbitrer untuk permintaan. Properti ini digunakan untuk memetakan permintaan input untuk respons output. Ketika ditentukan, `id` properti dalam objek respon diatur ke nilai ini. Jika Anda menghilangkan properti ini, konektor menghasilkan ID.

Wajib: `false`

Jenis: `string`

Pola yang valid: `.*`

Contoh masukan

```
{
    "request": {
        "message" : "{\"temp\":23.33}"
    },
    "id" : "req123"
}
```

Filter topik: `iotanalytics/channels/+/messages/binary/put`

Gunakan topik ini untuk mengirim pesan yang tidak diformat ke konektor dan secara dinamis menentukan saluran tujuan.

Data konektor tidak mengurai pesan input yang diterima pada topik ini. Ini memperlakukan mereka sebagai data biner. Sebelum mengirim pesan ke AWS IoT Analytics, konektor mengodekan dan memformatnya agar sesuai dengan `BatchPutMessage` persyaratan API:

- Konektor base64-encodes data mentah dan termasuk muatan dikodekan dalam permintaan `BatchPutMessage` outbound.
- Konektor menghasilkan dan menetapkan ID untuk setiap pesan input.

> **Note**
>
> Output respons konektor tidak menyertakan korelasi ID untuk pesan input ini.

Properti pesan

Tidak ada.

Filter topik: #

Gunakan topik ini untuk mengirim format pesan apa pun ke saluran default. Hal ini sangat berguna bila perangkat klien Anda menerbitkan ke topik tetap atau ketika Anda ingin mengirim data ke saluran default dari perangkat klien yang tidak dapat memproses data ke konektor format pesan yang didukung.

Anda menentukan sintaks topik dalam langganan yang Anda buat untuk menyambung konektor ini ke sumber data. Kami menyarankan Anda menentukan jalur topik alih-alih hanya menggunakan # wildcard dalam langganan Anda.

Data konektor tidak mengurai pesan yang diterbitkan untuk topik input ini. Semua pesan input diperlakukan sebagai data biner. Sebelum mengirim pesan ke AWS IoT Analytics, konektor mengodekan dan memformatnya agar sesuai dengan `BatchPutMessage` persyaratan API:

- Konektor base64-encodes data mentah dan termasuk muatan dikodekan dalam permintaan `BatchPutMessage` outbound.
- Konektor menghasilkan dan menetapkan ID untuk setiap pesan input.

> **Note**
>
> Output respons konektor tidak menyertakan korelasi ID untuk pesan input ini.

Properti pesan

Tidak ada.

Data output

Konektor ini menerbitkan informasi status sebagai data output pada topik MQTT. Informasi ini berisi respon dikembalikan oleh AWS IoT Analytics untuk setiap pesan input yang diterima dan dikirim ke AWS IoT Analytics.

Filter topik dalam langganan

`iotanalytics/messages/put/status`

Contoh output: Berhasil

```
{
    "response" : {
        "status" : "success"
```

```
    },
    "id" : "req123"
}
```

Contoh output: Gagal

```
{
    "response" : {
        "status" : "fail",
        "error" : "ResourceNotFoundException",
        "error_message" : "A resource with the specified name could not be found."
    },
    "id" : "req123"
}
```

> **Note**
>
> Jika konektor mendeteksi kesalahan yang dapat diulang (sebagai contoh, kesalahan koneksi), konektor mengulang lagi publikasinya dalam batch berikutnya. Backoff eksponensial ditangani oleh AWS SDK. Permintaan dengan kesalahan retryable ditambahkan kembali ke antrean saluran untuk penerbitan lebih lanjut sesuai dengan `IotAnalyticsQueueDropBehavior` parameter.

Contoh Penggunaan

Gunakan langkah-langkah tingkat tinggi berikut untuk mengatur contoh fungsi Lambda Python 3.7 yang dapat Anda gunakan untuk mencoba konektor.

> **Note**
>
> - Jika Anda menggunakan waktu aktif Python lainnya, Anda dapat membuat symlink dari Python3.x ke Python 3.7.
> - Topik Memulai dengan konektor (konsol) dan Memulai dengan konektor (CLI) berisi langkah-langkah terperinci yang menunjukkan cara mengonfigurasi dan men-deploy contoh konektor Notifikasi Twilio.

1. Pastikan Anda memenuhi persyaratan untuk konektor.

Untuk persyaratan peran grup, Anda harus mengonfigurasi peran untuk memberikan izin yang diperlukan dan memastikan peran telah ditambahkan ke grup. Untuk informasi lebih lanjut, lihat the section called “Mengelola peran grup (konsol)” atau the section called “Kelola peran grup (CLI)”.

2. Buat dan terbitkan fungsi Lambda yang mengirimkan data input ke konektor.

   Simpan kode contoh sebagai file PY. Unduh dan unzip AWS IoT Greengrass Core SDK for Python. Kemudian, buat paket zip yang berisi file PY dan folder `greengrasssdk` dalam tingkat root. Paket zip ini adalah paket deployment yang Anda unggah ke AWS Lambda.

   Setelah Anda membuat fungsi Lambda Python 3.7, terbitkankan versi fungsi dan buat alias.

3. Konfigurasikan grup Greengrass Anda.

   a. Tambahkan fungsi Lambda dengan aliasnya (direkomendasikan). Konfigurasikan siklus hidup Lambda sebagai berumur panjang (atau `"Pinned": true` dalam CLI).

   b. Tambahkan konektor dan konfigurasikan parameter.

   c. Tambahkan langganan yang mengizinkan konektor untuk menerima data input dan mengirim data output pada filter topik yang didukung.

      - Atur fungsi Lambda sebagai sumber, konektor sebagai target, dan gunakan filter topik input yang mendukung.
      - Atur konektor sebagai sumber, AWS IoT Core sebagai target, dan gunakan filter topik input yang mendukung. Anda menggunakan langganan ini untuk melihat pesan status dalam konsol AWS IoT tersebut.

4. Men-deploy grup.

5. Di konsol AWS IoT tersebut, pada halaman Tes ini, berlangganan ke topik data output untuk melihat pesan status dari konektor. Contoh fungsi Lambda yang berumur panjang dan mulai mengirim pesan segera setelah grup dalam-deploy.

   Setelah selesai pengujian, Anda dapat mengatur siklus hidup Lambda ke sesuai permintaan (atau `"Pinned": false` dalam CLI) dan men-deploy grup. Ini menghentikan fungsi dari mengirim pesan.

Contoh

Contoh fungsi Lambda berikut mengirimkan pesan input ke konektor.

```
import greengrasssdk
import time
import json

iot_client = greengrasssdk.client('iot-data')
send_topic = 'iotanalytics/channels/my_channel/messages/put'

def create_request_with_all_fields():
    return  {
        "request": {
            "message" : "{\"temp\":23.33}"
        },
        "id" : "req_123"
    }

def publish_basic_message():
    messageToPublish = create_request_with_all_fields()
    print("Message To Publish: ", messageToPublish)
    iot_client.publish(topic=send_topic,
        payload=json.dumps(messageToPublish))

publish_basic_message()

def lambda_handler(event, context):
    return
```

Batas

Konektor ini tunduk pada batas berikut.

- Semua batas yang diberlakukan olehAWS SDK for Python (Boto3) untukAWS IoT Analytics `batch_put_message`tindakan.
- Semua kuota yang dikenakan olehAWS IoT Analytics BatchPutMessageAPI. Untuk informasi selengkapnya, lihat Service Quotas untukAWS IoT Analytics di Referensi Umum AWS.
  - 100.000 pesan per detik per saluran.
  - 100 pesan per batch.
  - 128 KB per pesan.

  API ini menggunakan nama saluran (bukan saluran ARN), sehingga mengirim data ke saluran lintas wilayah atau lintas akun tidak didukung.

- Semua kuota yang dikenakan oleh AWS IoT Greengrass Core. Untuk informasi selengkapnya, lihat Service Quotas untukAWS IoT Greengrass inti di Referensi Umum AWS.

  Kuota berikut mungkin berlaku secara khusus:

  - Ukuran maksimum pesan yang dikirim oleh perangkat adalah 128 KB.
  - Ukuran antrian pesan maksimum dalam router core Greengrass adalah 2,5 MB.
  - Panjang maksimum string topik adalah 256 byte karakter yang dikodekan UTF-8.

Lisensi

Konektor IoT Analytics mencakup perangkat lunak/lisensi pihak ketiga berikut ini:

- AWS SDK for Python (Boto3)/Lisensi 2.0 Apache
- botocore/Apache License 2.0
- dateutil/PSF License
- docutils/BSD License, Lisensi Publik Umum (GPL) GNU, Lisensi Dasar Perangkat Lunak Python, Domain Publik
- jmespath/MIT License
- s3transfer/Apache License 2.0
- urllib3/Lisensi MIT

Konektor ini dirilis di bawah Perjanjian Lisensi Perangkat lunak core Greengrass.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi konektor.

Versi	Perubahan
4	Tambahkan `IsolationMode` parameter untuk mengonfigurasi mode kontainerisasi untuk konektor.
3	Memperbarui waktu aktif Lambda untuk Python 3.7, yang mengubah persyaratan waktu aktif.

Versi	Perubahan
2	Perbaiki untuk mengurangi pencatatan berlebihan.
1	Pelepasan .

Sebuah grup Greengrass dapat berisi hanya satu versi konektor pada suatu waktu. Untuk informasi lebih lanjut tentang pembaruan versi konektor, lihat the section called “Versi upgrade konektor”.

Lihat juga

- Integrasikan dengan layanan dan protokol menggunakan konektor
- the section called “Memulai dengan konektor (konsol)”
- the section called “Memulai dengan konektor (CLI)”
- Apa itu AWS IoT Analytics? dalam AWS IoT Analytics Panduan Penggguna

Konektor Adaptor Protokol IP Ethernet IoT

Konektor Adaptor Protokol IP Ethernet IoT mengumpulkan data dari perangkat lokal menggunakan protokol Ethernet/IP. Anda dapat menggunakan konektor ini untuk mengumpulkan data dari beberapa perangkat dan menerbitkannyanya ke `StreamManager` aliran pesan.

Anda juga dapat menggunakan konektor ini dengan IoT SiteWise konektor dan gateway IoT SiteWise Anda. Gateway Anda harus menyediakan konfigurasi untuk konektor. Untuk informasi selengkapnya, lihatMengkonfigurasi sumber Ethernet/IP (EIP)di IoT SiteWise panduan pengguna.

> **Note**
>
> Konektor ini berjalan dalam mode isolasi Tanpa kontainer ini, sehingga Anda dapat men-deploy ke grup AWS IoT Greengrass yang berjalan pada kontainer Docker.

Konektor ini memiliki versi berikut.

Versi	ARN
2 (direkomendasikan)	`arn:aws:greengrass:region::/connectors/IoTEIPProtocolAdaptor/versions/2`
1	`arn:aws:greengrass:region::/connectors/IoTEIPProtocolAdaptor/versions/1`

Untuk informasi tentang perubahan versi, lihat Changelog.

Persyaratan

Konektor ini memiliki persyaratan sebagai berikut:

Version 1 and 2

- AWS IoT Greengrass perangkat lunak Core v1.10.2 atau yang lebih baru.
- Pengelola streaming diaktifkan pada AWS IoT Greengrass grup.
- Java 8 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan `PATH` ini.
- Minimal 256 MB RAM tambahan. Persyaratan ini adalah tambahan persyaratan memori Core AWS IoT Greengrass ini.

> **Note**
>
> Konektor ini hanya tersedia dalam Wilayah berikut:
>
> - cn-north-1
> - ap-southeast-1
> - ap-southeast-2
> - eu-central-1
> - eu-west-1
> - us-east-1
> - us-west-2

Parameter Konektor

Konektor ini mendukung parameter berikut:

`LocalStoragePath`

Direktori diAWS IoT Greengrasshost bahwa IoT SiteWise konektor dapat menulis data persisten untuk. Direktori default adalah `/var/sitewise`.

Nama tampilan diAWS IoTKonsol: Jalur penyimpanan lokal

Wajib: `false`

Jenis: `string`

Pola yang valid: `^\s*$|\/.`

`ProtocolAdapterConfiguration`

Set konfigurasi kolektor Ethernet/IP yang konektor mengumpulkan data dari atau terhubung ke. Ini bisa menjadi daftar kosong.

Nama tampilan diAWS IoTKonsol: Konfigurasi Adaptor Protokol

Wajib: `true`

Jenis: Sebuah string JSON terbentuk yang mendefinisikan himpunan konfigurasi umpan balik yang didukung.

Berikut ini adalah contoh dari `ProtocolAdapterConfiguration`:

```
{
    "sources": [
        {
            "type": "EIPSource",
            "name": "TestSource",
            "endpoint": {
                "ipAddress": "52.89.2.42",
                "port": 44818
            },
            "destination": {
                "type": "StreamManager",
                "streamName": "MyOutput_Stream",
                "streamBufferSize": 10
```

```
            },
            "destinationPathPrefix": "EIPSource_Prefix",
            "propertyGroups": [
                {
                    "name": "DriveTemperatures",
                    "scanMode": {
                        "type": "POLL",
                        "rate": 10000
                    },
                    "tagPathDefinitions": [
                        {
                            "type": "EIPTagPath",
                            "path": "arrayREAL[0]",
                            "dstDataType": "double"
                        }
                    ]
                }
            ]
        }
    ]
}
```

Buat Contoh Konektor (AWS CLI)

Perintah CLI berikut membuat `ConnectorDefinition` dengan versi awal yang mengandung konektor Adaptor Protokol IP Ethernet IoT.

```
aws greengrass create-connector-definition --name MyGreengrassConnectors --initial-
version
'{
    "Connectors": [
        {
            "Id": "MyIoTEIPProtocolConnector",
            "ConnectorArn": "arn:aws:greengrass:region::/connectors/
IoTEIPProtocolAdaptor/versions/2",
            "Parameters": {
                "ProtocolAdaptorConfiguration": "{ \"sources\": [{ \"type
\": \"EIPSource\", \"name\": \"Source1\", \"endpoint\": { \"ipAddress\":
 \"54.245.77.218\", \"port\": 44818 }, \"destinationPathPrefix\": \"EIPConnector_Prefix
\", \"propertyGroups\": [{ \"name\": \"Values\", \"scanMode\": { \"type\": \"POLL\",
 \"rate\": 2000 }, \"tagPathDefinitions\": [{ \"type\": \"EIPTagPath\", \"path\":
 \"arrayREAL[0]\", \"dstDataType\": \"double\" }]}]}]}",
                "LocalStoragePath": "/var/MyIoTEIPProtocolConnectorState"
```

```
            }
        }
    ]
}'
```

> **Note**
>
> Fungsi Lambda dalam konektor mempunyai siklus hidup yang berumur panjang ini.

Data input

Konektor ini tidak menerima pesan MQTT sebagai data input.

Data output

Konektor ini menerbitkan data ke `StreamManager`. Anda harus mengonfigurasi aliran pesan tujuan. Pesan output dari struktur berikut:

```
{
    "alias": "string",
    "messages": [
        {
            "name": "string",
            "value": boolean|double|integer|string,
            "timestamp": number,
            "quality": "string"
        }
    ]
}
```

Lisensi

Konektor Adaptor Protokol IP Ethernet IoT mencakup perangkat lunak/lisensi pihak ketiga berikut:

- Klien Ethernet/IP
- MapDB
- Elsa

Konektor ini dirilis di bawah Perjanjian Lisensi Perangkat lunak core Greengrass.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi konektor.

Versi	Perubahan	Tanggal
2	Versi ini berisi perbaikan bug.	23 Desember 2021
1	Pelepasan .	15 Desember 2020

Sebuah grup Greengrass dapat berisi hanya satu versi konektor pada suatu waktu. Untuk informasi lebih lanjut tentang pembaruan versi konektor, lihat the section called “Versi upgrade konektor”.

Lihat juga

- Integrasikan dengan layanan dan protokol menggunakan konektor
- the section called “Memulai dengan konektor (konsol)”
- the section called “Memulai dengan konektor (CLI)”

Konektor IoT SiteWise

SiteWise Konektor IoT mengirimkan data perangkat dan peralatan lokal ke properti aset di. AWS IoT SiteWise Anda dapat menggunakan konektor ini untuk mengumpulkan data dari beberapa server OPC-UA dan mempublikasikannya ke IoT. SiteWise Konektor mengirimkan data ke properti aset dalam arus Akun AWS dan Wilayah.

> **Note**
>
> IoT SiteWise adalah layanan yang dikelola sepenuhnya yang mengumpulkan, memproses, dan memvisualisasikan data dari perangkat dan peralatan industri. Anda dapat mengonfigurasi properti aset yang memproses data mentah yang dikirim dari konektor ini ke properti pengukuran aset Anda. Sebagai contoh, Anda dapat menentukan properti transformasi yang mengkonversi Celcius titik data suhu perangkat ke Fahrenheit, atau Anda dapat menentukan properti metrik yang menghitung suhu per jam rata-rata. Untuk informasi lebih lanjut, lihat Apa AWS IoT SiteWise? dalam AWS IoT SiteWise Panduan Pengguna.

Konektor mengirimkan data ke IoT SiteWise dengan jalur aliran data OPC-UA yang dikirim dari server OPC-UA. Sebagai contoh, jalur aliran data `/company/windfarm/3/turbine/7/temperature` mungkin mewakili sensor suhu turbin #7 dalam ladang angin #3. Jika AWS IoT Greengrass core kehilangan koneksi ke internet, konektor data cache sampai berhasil terhubung ke AWS Cloud. Anda dapat mengonfigurasi ukuran buffer disk maksimum yang digunakan untuk data caching. Jika ukuran cache melebihi ukuran buffer disk maksimum, konektor membuang data terlama dari antrian.

Setelah mengonfigurasi dan menerapkan konektor SiteWise IoT, Anda dapat menambahkan gateway dan sumber OPC-UA di konsol IoT. SiteWise Saat mengonfigurasi sumber di konsol, Anda dapat memfilter atau mengawali jalur aliran data OPC-UA yang dikirim oleh konektor IoT. SiteWise Untuk instruksi untuk menyelesaikan pengaturan gateway dan sumber, lihat Menambahkan gateway dalam AWS IoT SiteWise Panduan Pengguna.

IoT SiteWise menerima data hanya dari aliran data yang telah Anda petakan ke properti pengukuran aset IoT. SiteWise Untuk memetakan aliran data ke properti aset, Anda dapat mengatur alias properti untuk setara dengan jalur aliran data OPC-UA. Untuk mempelajari tentang mendefinisikan model aset dan menciptakan aset, lihat Pemodelan aset industri dalam AWS IoT SiteWise Panduan Pengguna.

> **Catatan**
>
> Anda dapat menggunakan pengelola aliran untuk mengunggah data ke IoT SiteWise dari sumber selain server OPC-UA. Stream manager juga menyediakan dukungan disesuaikan untuk ketahanan dan manajemen bandwidth. Untuk informasi selengkapnya, lihat Mengelola aliran data.
>
> Konektor ini berjalan dalam mode isolasi Tanpa kontainer ini, sehingga Anda dapat men-deploy ke grup Greengrass yang berjalan pada kontainer Docker.

Konektor ini memiliki versi berikut.

Versi	ARN
12 (disarankan)	`arn:aws:greengrass:  region::/connectors/IoTSiteWise/versions/ 12`

Versi	ARN
11	arn:aws:greengrass:*region*::/connectors/IoTSiteWise/versions/11
10	arn:aws:greengrass:*region*::/connectors/IoTSiteWise/versions/10
9	arn:aws:greengrass:*region*::/connectors/IoTSiteWise/versions/9
8	arn:aws:greengrass:*region*::/connectors/IoTSiteWise/versions/8
7	arn:aws:greengrass:*region*::/connectors/IoTSiteWise/versions/7
6	arn:aws:greengrass:*region*::/connectors/IoTSiteWise/versions/6
5	arn:aws:greengrass:*region*::/connectors/IoTSiteWise/versions/5
4	arn:aws:greengrass:*region*::/connectors/IoTSiteWise/versions/4
3	arn:aws:greengrass:*region*::/connectors/IoTSiteWise/versions/3

Versi	ARN
2	`arn:aws:greengrass:  region::/connectors/IoTSiteWise/versions/ 2`
1	`arn:aws:greengrass:  region::/connectors/IoTSiteWise/versions/ 1`

Untuk informasi tentang perubahan versi, lihat Changelog.

Persyaratan

Konektor ini memiliki persyaratan sebagai berikut:

Version 9, 10, 11, and 12

> **Important**
>
> Versi ini memperkenalkan persyaratan baru: AWS IoT Greengrass Perangkat lunak Core v1.10.2 dan pengelola aliran.

- AWS IoT Greengrass perangkat lunak Core v1.10.2.
- Stream manager diaktifkan pada grup Greengrass.
- Java 8 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.
- Konektor ini hanya dapat digunakan di Wilayah Amazon Web Services di mana keduanya AWS IoT Greengrassdan IoT SiteWise didukung.
- Sebuah kebijakan IAM ditambahkan ke peran grup Greengrass. Peran ini mengizinkan akses grup AWS IoT Greengrass ke tindakan `iotsitewise:BatchPutAssetPropertyValue` pada aset root target dan turunannya, seperti yang ditunjukkan dalam contoh berikut. Anda dapat menghapus `Condition` dari kebijakan untuk memungkinkan konektor mengakses semua aset IoT SiteWise Anda.

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

```
    "Statement": [
        {
             "Effect": "Allow",
             "Action": "iotsitewise:BatchPutAssetPropertyValue",
             "Resource": "*",
             "Condition": {
                 "StringLike": {
                     "iotsitewise:assetHierarchyPath": [
                         "/root node asset ID",
                         "/root node asset ID/*"
                     ]
                 }
             }
        }
    ]
}
```

Untuk informasi lebih lanjut, lihat Menambahkan dan menghapus kebijakan IAM dalam Panduan Pengguna IAM.

Versions 6, 7, and 8

> **Important**
>
> Versi ini memperkenalkan persyaratan baru: AWS IoT Greengrass Perangkat lunak Core v1.10.0 dan pengelola aliran.

- AWS IoT Greengrass perangkat lunak Core v1.10.0.
- Stream manager diaktifkan pada grup Greengrass.
- Java 8 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.
- Konektor ini hanya dapat digunakan di Wilayah Amazon Web Services di mana keduanya AWS IoT Greengrassdan IoT SiteWise didukung.
- Sebuah kebijakan IAM ditambahkan ke peran grup Greengrass. Peran ini mengizinkan akses grup AWS IoT Greengrass ke tindakan `iotsitewise:BatchPutAssetPropertyValue` pada aset root target dan turunannya, seperti yang ditunjukkan dalam contoh berikut. Anda dapat menghapus `Condition` dari kebijakan untuk memungkinkan konektor mengakses semua aset IoT SiteWise Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
             "Effect": "Allow",
             "Action": "iotsitewise:BatchPutAssetPropertyValue",
             "Resource": "*",
             "Condition": {
                 "StringLike": {
                     "iotsitewise:assetHierarchyPath": [
                         "/root node asset ID",
                         "/root node asset ID/*"
                     ]
                 }
             }
        }
    ]
}
```

Untuk informasi lebih lanjut, lihat Menambahkan dan menghapus kebijakan IAM dalam Panduan Pengguna IAM.

Version 5

- AWS IoT Greengrass perangkat lunak Core v1.9.4.
- Java 8 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.
- Konektor ini hanya dapat digunakan di Wilayah Amazon Web Services di mana keduanya AWS IoT Greengrassdan IoT SiteWise didukung.
- Sebuah kebijakan IAM ditambahkan ke peran grup Greengrass. Peran ini mengizinkan akses grup AWS IoT Greengrass ke tindakan `iotsitewise:BatchPutAssetPropertyValue` pada aset root target dan turunannya, seperti yang ditunjukkan dalam contoh berikut. Anda dapat menghapus `Condition` dari kebijakan untuk memungkinkan konektor mengakses semua aset IoT SiteWise Anda.

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

```
            "Action": "iotsitewise:BatchPutAssetPropertyValue",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "iotsitewise:assetHierarchyPath": [
                        "/root node asset ID",
                        "/root node asset ID/*"
                    ]
                }
            }
        }
    ]
}
```

Untuk informasi lebih lanjut, lihat Menambahkan dan menghapus kebijakan IAM dalam Panduan Pengguna IAM.

Version 4

- AWS IoT Greengrass perangkat lunak Core v1.10.0.
- Java 8 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.
- Konektor ini hanya dapat digunakan di Wilayah Amazon Web Services di mana keduanya AWS IoT Greengrassdan IoT SiteWise didukung.
- Sebuah kebijakan IAM ditambahkan ke peran grup Greengrass. Peran ini mengizinkan akses grup AWS IoT Greengrass ke tindakan `iotsitewise:BatchPutAssetPropertyValue` pada aset root target dan turunannya, seperti yang ditunjukkan dalam contoh berikut. Anda dapat menghapus `Condition` dari kebijakan untuk memungkinkan konektor mengakses semua aset IoT SiteWise Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iotsitewise:BatchPutAssetPropertyValue",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "iotsitewise:assetHierarchyPath": [
                        "/root node asset ID",
```

```
                        "/root node asset ID/*"
                    ]
                }
            }
        ]
}
```

Untuk informasi lebih lanjut, lihat Menambahkan dan menghapus kebijakan IAM dalam Panduan Pengguna IAM.

Version 3

- AWS IoT Greengrass perangkat lunak Core v1.9.4.
- Java 8 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.
- Konektor ini hanya dapat digunakan di Wilayah Amazon Web Services di mana keduanya AWS IoT Greengrassdan IoT SiteWise didukung.
- Sebuah kebijakan IAM ditambahkan ke peran grup Greengrass. Peran ini mengizinkan akses grup AWS IoT Greengrass ke tindakan `iotsitewise:BatchPutAssetPropertyValue` pada aset root target dan turunannya, seperti yang ditunjukkan dalam contoh berikut. Anda dapat menghapus `Condition` dari kebijakan untuk memungkinkan konektor mengakses semua aset IoT SiteWise Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
             "Effect": "Allow",
             "Action": "iotsitewise:BatchPutAssetPropertyValue",
             "Resource": "*",
             "Condition": {
                 "StringLike": {
                     "iotsitewise:assetHierarchyPath": [
                         "/root node asset ID",
                         "/root node asset ID/*"
                     ]
                 }
             }
        }
    ]
```

```
}
```

Untuk informasi lebih lanjut, lihat Menambahkan dan menghapus kebijakan IAM dalam Panduan Pengguna IAM.

Versions 1 and 2

- AWS IoT Greengrass perangkat lunak Core v1.9.4.
- Java 8 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.
- Konektor ini hanya dapat digunakan di Wilayah Amazon Web Services di mana keduanya AWS IoT Greengrassdan IoT SiteWise didukung.
- Kebijakan IAM ditambahkan ke peran grup Greengrass yang mengizinkan akses ke AWS IoT Core dan `iotsitewise:BatchPutAssetPropertyValue` tindakan pada aset root target dan anak-anaknya, seperti yang ditunjukkan dalam contoh berikut. Anda dapat menghapus `Condition` dari kebijakan untuk memungkinkan konektor mengakses semua aset IoT SiteWise Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
             "Effect": "Allow",
             "Action": "iotsitewise:BatchPutAssetPropertyValue",
             "Resource": "*",
             "Condition": {
                 "StringLike": {
                     "iotsitewise:assetHierarchyPath": [
                         "/root node asset ID",
                         "/root node asset ID/*"
                     ]
                 }
             }
        },
        {
            "Effect": "Allow",
            "Action": [
                 "iot:Connect",
                 "iot:DescribeEndpoint",
                 "iot:Publish",
                 "iot:Receive",
```

```
                "iot:Subscribe"
            ],
            "Resource": "*"
        }
    ]
}
```

Untuk informasi lebih lanjut, lihat Menambahkan dan menghapus izin identitas IAM dalam Panduan Pengguna IAM.

Parameter

Versions 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, and 12

`SiteWiseLocalStoragePath`

Direktori pada AWS IoT Greengrass host tempat SiteWise konektor IoT dapat menulis data persisten. Default ke `/var/sitewise`.

Nama tampilan pada konsol AWS IoT tersebut: Jalur penyimpanan lokal

Wajib: `false`

Jenis: `string`

Pola yang valid: `^\s*$|\/.`

`AWSSecretsArnList`

Daftar rahasia dalam AWS Secrets Manager bahwa masing-masing berisi nama pengguna OPC-UA dan kata sandi kunci-nilai pasangan. Setiap rahasia harus menjadi rahasia jenis pasangan yang bernilai kunci.

Nama tampilan pada konsol AWS IoT tersebut: Daftar ARN untuk rahasia nama pengguna/kata sandi OPC-UA

Wajib: `false`

Jenis: `JsonArrayOfStrings`

Pola yang valid: `\[( ?,? ?\"(arn:(aws(-[a-z]+)*):secretsmanager:[a-z0-9\\-]+:[0-9]{12}:secret:([a-zA-Z0-9\\\\]+\/)*[a-zA-Z0-9\/_+=,.@\\-]+-[a-zA-Z0-9]+)*\")*\]`

`MaximumBufferSize`

Ukuran maksimum dalam GB untuk penggunaan SiteWise disk IoT. Default untuk 10GB.

Nama tampilan pada konsol AWS IoT tersebut: Ukuran maksimum buffer disk

Wajib: `false`

Jenis: `string`

Pola yang valid: `^\s*$|[0-9]+`

Version 1

`SiteWiseLocalStoragePath`

Direktori pada AWS IoT Greengrass host tempat SiteWise konektor IoT dapat menulis data persisten. Default ke `/var/sitewise`.

Nama tampilan pada konsol AWS IoT tersebut: Jalur penyimpanan lokal

Wajib: `false`

Jenis: `string`

Pola yang valid: `^\s*$|\/.`

`SiteWiseOpcuaUserIdentityTokenSecretArn`

Rahasia dalam AWS Secrets Manager yang berisi nama pengguna OPC-UA dan kata sandi kunci-nilai pasangan. Rahasia ini harus menjadi rahasia jenis pasangan kunci-nilai.

Nama tampilan pada konsol AWS IoT tersebut: ARN rahasia nama pengguna/kata sandi OPC-UA

Wajib: `false`

Jenis: `string`

Pola yang valid: `^$|arn:(aws(-[a-z]+)*):secretsmanager:[a-z0-9\\-]+:[0-9]{12}:secret:([a-zA-Z0-9\\\\]+/)*[a-zA-Z0-9/_+=,.@\\-]+-[a-zA-Z0-9]+`

`SiteWiseOpcuaUserIdentityTokenSecretArn-ResourceId`

Sumber daya rahasia dalam AWS IoT Greengrass grup yang mereferensikan rahasia nama pengguna dan kata sandi OPC-UA.

Nama tampilan pada konsol AWS IoT tersebut: sumber daya rahasia nama pengguna/kata sandi OPC-UA

Wajib: `false`

Jenis: `string`

Pola yang valid: `^$|.+`

`MaximumBufferSize`

Ukuran maksimum dalam GB untuk penggunaan SiteWise disk IoT. Default untuk 10GB.

Nama tampilan pada konsol AWS IoT tersebut: Ukuran maksimum buffer disk

Wajib: `false`

Jenis: `string`

Pola yang valid: `^\s*$|[0-9]+`

Buat Contoh Konektor (AWS CLI)

AWS CLIPerintah berikut membuat `ConnectorDefinition` dengan versi awal yang berisi konektor IoT SiteWise .

```
aws greengrass create-connector-definition --name MyGreengrassConnectors --initial-
version '{
    "Connectors": [
        {
            "Id": "MyIoTSiteWiseConnector",
            "ConnectorArn": "arn:aws:greengrass:region::/connectors/IoTSiteWise/
versions/11"
        }
    ]
}'
```

> **Note**
>
> Fungsi-fungsi Lambda dalam konektor mempunyai siklus hidup yang berumur panjang ini.

Di konsol AWS IoT Greengrass tersebut, Anda dapat menambahkan konektor dari halaman grup Konektor ini. Untuk informasi lebih lanjut, lihat the section called “Memulai dengan konektor (konsol)”.

Data input

Konektor ini tidak menerima pesan MQTT sebagai data input.

Data output

Konektor ini tidak menerbitkankan pesan MQTT sebagai Data output.

Batas

Konektor ini tunduk pada semua batasan berikut yang diberlakukan oleh IoT SiteWise, termasuk yang berikut ini. Untuk informasi lebih lanjut, lihat AWS IoT SiteWisetitik akhir dan kuota di. Referensi Umum AWS

- Jumlah maksimum gateway per Akun AWS.
- Jumlah maksimum sumber OPC-UA per gateway.
- Tingkat maksimum titik data timestamp-quality-value (TQV) yang disimpan per. Akun AWS
- Tingkat maksimum titik data TQV yang dipertahankan per properti aset.

Lisensi

Version 9, 10, 11, and 12

SiteWise Konektor IoT mencakup perangkat lunak/lisensi pihak ketiga berikut:

- MapDB
- Elsa
- Gerhana Milo

Konektor ini dirilis di bawah Perjanjian Lisensi Perangkat lunak core Greengrass.

Versions 6, 7, and 8

SiteWise Konektor IoT mencakup perangkat lunak/lisensi pihak ketiga berikut:

- Milo / EDL 1.0

Konektor ini dirilis di bawah Perjanjian Lisensi Perangkat lunak core Greengrass.

Versions 1, 2, 3, 4, and 5

SiteWise Konektor IoT mencakup perangkat lunak/lisensi pihak ketiga berikut:

- Milo / EDL 1.0
- Chronicle-Queue / Lisensi 2.0 Apache

Konektor ini dirilis di bawah Perjanjian Lisensi Perangkat lunak core Greengrass.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi konektor.

Versi	Perubahan	Tanggal
12	• Versi ini berisi perbaikan bug.	Desember 22, 2021
11	• Support untuk string yang berisi karakter tersembunyi atau unprintable. Karakter tersembunyi dan unprintable secara otomatis dihapus sebelum string dikirim ke AWS Cloud. • Memperbaiki masalah yang menyebabkan SiteWise gateway IoT mencoba ulang permintaan yang tidak valid secara tak terbatas.	24 Maret 2021

Versi	Perubahan	Tanggal
	• Memperbaiki masalah yang menyebabkan pos pemeriksaan rusak saat SiteWise gateway IoT terhubung ke sumber data frekuensi tinggi. • Peningkatan pesan kesalahan untuk membantu menyelesaikan masalah konfigurasi gateway.	
10	Mengonfigurasi `StreamManager` untuk meningkatkan penanganan ketika koneksi sumber hilang dan didirikan kembali. Versi ini juga menerima nilai OPC-UA dengan sebuah `ServerTimestamp` ketika tidak `SourceTimestamp` tersedia.	22 Januari 2021
9	Support diluncurkan untuk tujuan pengaliran `StreamManager` Greengrass kustom, deadbanding OPC-UA, mode pemindaian khusus dan tingkat pemindaian kustom. Juga termasuk peningkatan kinerja selama pembaruan konfigurasi yang dibuat dari gateway IoT SiteWise .	15 Desember 2020

Versi	Perubahan	Tanggal
8	Peningkatan stabilitas ketika konektor mengalami konektivitas jaringan intermiten.	19 November 2020
7	Memperbaiki masalah dengan metrik gateway.	14 Agustus 2020
6	Menambahkan dukungan untuk CloudWatch metrik dan penemuan otomatis tag OPC-UA baru. Versi ini membutuhkan pengelola aliran dan Perangkat lunak Core AWS IoT Greengrass v1.10.0 atau yang lebih tinggi.	29 April 2020
5	Memperbaiki masalah kompatibilitas dengan Perangkat lunak Core AWS IoT Greengrass v1.9.4.	12 Februari 2020
4	Memperbaiki masalah dengan rekoneksi server OPC-UA.	7 Februari 2020
3	Menghapus `iot:*` persyaratan izin.	17 Desember 2019
2	Ditambahkan dukungan untuk beberapa sumber daya rahasia OPC-UA.	10 Desember 2019
1	Pelepasan awal.	2 Desember 2019

Sebuah grup Greengrass dapat berisi hanya satu versi konektor pada suatu waktu. Untuk informasi lebih lanjut tentang pembaruan versi konektor, lihat the section called “Versi upgrade konektor”.

Lihat juga

- Integrasikan dengan layanan dan protokol menggunakan konektor
- the section called “Memulai dengan konektor (konsol)”
- the section called “Memulai dengan konektor (CLI)”
- Lihat topik berikut dalam Panduan PenggunaAWS IoT SiteWise:
  - Apa itu AWS IoT SiteWise?
  - Menggunakan gateway
  - CloudWatch Metrik gerbang
  - Memecahkan masalah gateway IoT SiteWise

Kinesis Firehose

Konektor Kinesis Firehose menerbitkan data melalui aliran pengiriman Amazon Data Firehose ke tujuan seperti Amazon S3, Amazon Redshift, atau Amazon Service. OpenSearch

Konektor ini adalah produsen data untuk aliran pengiriman Kinesis. Ia menerima data input pada topik MQTT, dan mengirimkan data ke aliran pengiriman tertentu. Aliran pengiriman kemudian mengirimkan catatan data ke tujuan yang dikonfigurasi (sebagai contoh, sebuah bucket S3).

Konektor ini memiliki versi berikut.

Versi	ARN
5	`arn:aws:greengrass:  region::/connectors/KinesisFirehose/versions/5`
4	`arn:aws:greengrass:  region::/connectors/KinesisFirehose/versions/4`
3	`arn:aws:greengrass:  region::/connectors/KinesisFirehose/versions/3`

Versi	ARN
2	`arn:aws:greengrass:  region::/connectors/KinesisFirehose/versions/2`
1	`arn:aws:greengrass:  region::/connectors/KinesisFirehose/versions/1`

Untuk informasi tentang perubahan versi, lihat Changelog.

Persyaratan

Konektor ini memiliki persyaratan sebagai berikut:

Version 4 - 5

- AWS IoT Greengrass Perangkat lunak inti v1.9.3 atau yang lebih baru.
- Python versi 3.7 atau 3.8 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.

  > **Note**
  >
  > Untuk menggunakan Python 3.8, jalankan perintah berikut untuk membuat link simbolik dari folder instalasi default Python 3.7 ke binari Python 3.8 yang diinstal.
  >
  > ```
  > sudo ln -s path-to-python-3.8/python3.8 /usr/bin/python3.7
  > ```
  >
  > Ini mengonfigurasi perangkat Anda untuk memenuhi persyaratan Python untuk AWS IoT Greengrass.

- Aliran pengiriman Kinesis yang dikonfigurasi. Untuk informasi selengkapnya, lihat Membuat aliran pengiriman Amazon Data Firehose di Panduan Pengembang Amazon Kinesis Firehose.
- Peran grup Greengrass dikonfigurasi untuk mengizinkan `firehose:PutRecord` dan tindakan `firehose:PutRecordBatch` dalam aliran pengiriman target, seperti yang ditunjukkan dalam contoh kebijakan IAM berikut.

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"Stmt1528133056761",
            "Action":[
                "firehose:PutRecord",
                "firehose:PutRecordBatch"
            ],
            "Effect":"Allow",
            "Resource":[
                "arn:aws:firehose:region:account-id:deliverystream/stream-name"
            ]
        }
    ]
}
```

Konektor ini mengizinkan Anda untuk secara dinamis menimpa aliran pengiriman default dalam muatan pesan input. Jika implementasi Anda menggunakan fitur ini, kebijakan IAM harus mencakup semua aliran target sebagai sumber daya. Anda dapat memberikan akses terperinci atau bersyarat ke sumber daya (sebagai contoh, dengan menggunakan skema penamaan wildcard *).

Untuk persyaratan peran grup, Anda harus mengonfigurasi peran untuk memberikan izin yang diperlukan dan memastikan peran telah ditambahkan ke grup. Untuk informasi lebih lanjut, lihat the section called “Mengelola peran grup (konsol)” atau the section called “Kelola peran grup (CLI)”.

Versions 2 - 3

- AWS IoT Greengrass Perangkat lunak inti v1.7 atau yang lebih baru.
- Python versi 2.7 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.
- Aliran pengiriman Kinesis yang dikonfigurasi. Untuk informasi selengkapnya, lihat Membuat aliran pengiriman Amazon Data Firehose di Panduan Pengembang Amazon Kinesis Firehose.
- Peran grup Greengrass dikonfigurasi untuk mengizinkan `firehose:PutRecord` dan tindakan `firehose:PutRecordBatch` dalam aliran pengiriman target, seperti yang ditunjukkan dalam contoh kebijakan IAM berikut.

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"Stmt1528133056761",
            "Action":[
                "firehose:PutRecord",
                "firehose:PutRecordBatch"
            ],
            "Effect":"Allow",
            "Resource":[
                "arn:aws:firehose:region:account-id:deliverystream/stream-name"
            ]
        }
    ]
}
```

Konektor ini mengizinkan Anda untuk secara dinamis menimpa aliran pengiriman default dalam muatan pesan input. Jika implementasi Anda menggunakan fitur ini, kebijakan IAM harus mencakup semua aliran target sebagai sumber daya. Anda dapat memberikan akses terperinci atau bersyarat ke sumber daya (sebagai contoh, dengan menggunakan skema penamaan wildcard *).

Untuk persyaratan peran grup, Anda harus mengonfigurasi peran untuk memberikan izin yang diperlukan dan memastikan peran telah ditambahkan ke grup. Untuk informasi lebih lanjut, lihat the section called “Mengelola peran grup (konsol)” atau the section called “Kelola peran grup (CLI)”.

Version 1

- AWS IoT Greengrass Perangkat lunak inti v1.7 atau yang lebih baru.
- Python versi 2.7 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.
- Aliran pengiriman Kinesis yang dikonfigurasi. Untuk informasi selengkapnya, lihat Membuat aliran pengiriman Amazon Data Firehose di Panduan Pengembang Amazon Kinesis Firehose.
- Peran grup Greengrass dikonfigurasi untuk mengizinkan `firehose:PutRecord` tindakan dalam aliran pengiriman target, seperti yang ditunjukkan dalam kebijakan IAM berikut.

```
{
```

```
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"Stmt1528133056761",
            "Action":[
                "firehose:PutRecord"
            ],
            "Effect":"Allow",
            "Resource":[
                "arn:aws:firehose:region:account-id:deliverystream/stream-name"
            ]
        }
    ]
}
```

Konektor ini mengizinkan Anda untuk secara dinamis menimpa aliran pengiriman default dalam muatan pesan input. Jika implementasi Anda menggunakan fitur ini, kebijakan IAM harus mencakup semua aliran target sebagai sumber daya. Anda dapat memberikan akses terperinci atau bersyarat ke sumber daya (sebagai contoh, dengan menggunakan skema penamaan wildcard *).

Untuk persyaratan peran grup, Anda harus mengonfigurasi peran untuk memberikan izin yang diperlukan dan memastikan peran telah ditambahkan ke grup. Untuk informasi lebih lanjut, lihat the section called “Mengelola peran grup (konsol)” atau the section called “Kelola peran grup (CLI)”.

Parameter Konektor

Konektor ini menyediakan parameter berikut:

Versions 5

`DefaultDeliveryStreamArn`

ARN dari aliran pengiriman Firehose default untuk mengirim data ke. Aliran tujuan dapat diganti oleh properti `delivery_stream_arn` dalam muatan pesan input.

> **Note**
>
> Peran grup harus mengizinkan tindakan yang sesuai pada semua aliran pengiriman target. Untuk informasi selengkapnya, lihat the section called “Persyaratan”.

Nama tampilan di AWS IoT konsol: Arn aliran pengiriman default

Wajib: `true`

Jenis: `string`

Pola yang valid: `arn:aws:firehose:([a-z]{2}-[a-z]+-\d{1}):(\d{12}):deliverystream/([a-zA-Z0-9_\-.]+)$`

`DeliveryStreamQueueSize`

Jumlah maksimum data untuk mempertahankan dalam memori sebelum catatan baru untuk aliran pengiriman yang sama ditolak. Nilai minimumnya adalah 2000.

Nama tampilan di AWS IoT konsol: Jumlah maksimum catatan untuk buffer (per aliran)

Wajib: `true`

Jenis: `string`

Pola yang valid: `^([2-9]\\d{3}|[1-9]\\d{4,})$`

`MemorySize`

Jumlah memori (dalam KB) untuk mengalokasikan ke konektor ini.

Nama tampilan di AWS IoT konsol: Ukuran memori

Wajib: `true`

Jenis: `string`

Pola yang valid: `^[0-9]+$`

`PublishInterval`

Interval (dalam detik) untuk menerbitkan catatan ke Firehose. Untuk menonaktifkan batching, atur nilai ini ke 0.

Nama tampilan di AWS IoT konsol: Publikasikan interval

Wajib: `true`

Jenis: `string`

Nilai yang valid: `0 - 900`

Pola yang valid: `[0-9]|[1-9]\\d|[1-9]\\d\\d|900`

`IsolationMode`

Mode kontainerisasi untuk konektor ini. Defaultnya adalah`GreengrassContainer`, yang berarti konektor berjalan di lingkungan runtime yang terisolasi di dalam AWS IoT Greengrass container.

> **Note**
>
> Pengaturan kontainerisasi default untuk grup tidak berlaku untuk konektor.

Nama tampilan di AWS IoT konsol: Mode isolasi kontainer

Wajib: `false`

Jenis: `string`

Nilai yang valid: `GreengrassContainer` or `NoContainer`

Pola yang valid: `^NoContainer$|^GreengrassContainer$`

Versions 2 - 4

`DefaultDeliveryStreamArn`

ARN dari aliran pengiriman Firehose default untuk mengirim data ke. Aliran tujuan dapat diganti oleh properti `delivery_stream_arn` dalam muatan pesan input.

> **Note**
>
> Peran grup harus mengizinkan tindakan yang sesuai pada semua aliran pengiriman target. Untuk informasi selengkapnya, lihat the section called “Persyaratan”.

Nama tampilan di AWS IoT konsol: Arn aliran pengiriman default

Wajib: `true`

Jenis: `string`

Pola yang valid: `arn:aws:firehose:([a-z]{2}-[a-z]+-\d{1}):(\d{12}):deliverystream/([a-zA-Z0-9_\-.]+)$`

`DeliveryStreamQueueSize`

Jumlah maksimum data untuk mempertahankan dalam memori sebelum catatan baru untuk aliran pengiriman yang sama ditolak. Nilai minimumnya adalah 2000.

Nama tampilan di AWS IoT konsol: Jumlah maksimum catatan untuk buffer (per aliran)

Wajib: `true`

Jenis: `string`

Pola yang valid: `^([2-9]\\d{3}|[1-9]\\d{4,})$`

`MemorySize`

Jumlah memori (dalam KB) untuk mengalokasikan ke konektor ini.

Nama tampilan di AWS IoT konsol: Ukuran memori

Wajib: `true`

Jenis: `string`

Pola yang valid: `^[0-9]+$`

`PublishInterval`

Interval (dalam detik) untuk menerbitkan catatan ke Firehose. Untuk menonaktifkan batching, atur nilai ini ke 0.

Nama tampilan di AWS IoT konsol: Publikasikan interval

Wajib: `true`

Jenis: `string`

Nilai yang valid: `0 - 900`

Pola yang valid: `[0-9]|[1-9]\\d|[1-9]\\d\\d|900`

Version 1

`DefaultDeliveryStreamArn`

ARN dari aliran pengiriman Firehose default untuk mengirim data ke. Aliran tujuan dapat diganti oleh properti `delivery_stream_arn` dalam muatan pesan input.

> **Note**
>
> Peran grup harus mengizinkan tindakan yang sesuai pada semua aliran pengiriman target. Untuk informasi selengkapnya, lihat the section called “Persyaratan”.

Nama tampilan di AWS IoT konsol: Arn aliran pengiriman default

Wajib: `true`

Jenis: `string`

Pola yang valid: `arn:aws:firehose:([a-z]{2}-[a-z]+-\d{1}):(\d{12}):deliverystream/([a-zA-Z0-9_\-.]+)$`

Example

Buat Contoh Konektor (AWS CLI)

Perintah CLI berikut membuat `ConnectorDefinition` dengan versi awal yang mengandungi konektor.

```
aws greengrass create-connector-definition --name MyGreengrassConnectors --initial-
version '{
    "Connectors": [
        {
            "Id": "MyKinesisFirehoseConnector",
            "ConnectorArn": "arn:aws:greengrass:region::/connectors/KinesisFirehose/
versions/5",
            "Parameters": {
```

```
                "DefaultDeliveryStreamArn": "arn:aws:firehose:region:account-
id:deliverystream/stream-name",
                "DeliveryStreamQueueSize": "5000",
                "MemorySize": "65535",
                "PublishInterval": "10",
                "IsolationMode" : "GreengrassContainer"
            }
        }
    ]
}'
```

Di AWS IoT Greengrass konsol, Anda dapat menambahkan konektor dari halaman Konektor grup. Untuk informasi selengkapnya, lihat the section called “Memulai dengan konektor (konsol)”.

Data input

Konektor ini menerima konten stream pada topik MQTT, dan kemudian mengirimkan konten ke aliran pengiriman target. Ia menerima dua jenis data input:

- Data JSON pada `kinesisfirehose/message` Topik.
- Data biner pada `kinesisfirehose/message/binary/#` Topik.

Versions 2 - 5

Filter topik: `kinesisfirehose/message`

Gunakan topik ini untuk mengirim pesan yang berisi data JSON.

Properti pesan

`request`

Data yang akan dikirim ke aliran pengiriman dan aliran pengiriman target, jika berbeda dari aliran default.

Wajib: `true`

Jenis: `object` yang mencakup properti berikut:

`data`

Data untuk mengirim ke aliran pengiriman.

Wajib: `true`

Jenis: `string`

`delivery_stream_arn`

ARN dari target aliran pengiriman Kinesis. Sertakan properti ini untuk menimpa aliran pengiriman default.

Wajib: `false`

Jenis: `string`

Pola yang valid: `arn:aws:firehose:([a-z]{2}-[a-z]+-\d{1}):(\d{12}):deliverystream/([a-zA-Z0-9_\-.]+)$`

`id`

ID arbitrer untuk permintaan. Properti ini digunakan untuk memetakan permintaan input untuk respons output. Ketika ditentukan, `id` properti dalam objek respon diatur ke nilai ini. Jika Anda tidak menggunakan fitur ini, Anda dapat menghilangkan properti ini atau menentukan string kosong.

Wajib: `false`

Jenis: `string`

Pola yang valid: `.*`

Contoh masukan

```
{
     "request": {
        "delivery_stream_arn": "arn:aws:firehose:region:account-
id:deliverystream/stream2-name",
        "data": "Data to send to the delivery stream."
     },
     "id": "request123"
}
```

Filter topik: `kinesisfirehose/message/binary/#`

Gunakan topik ini untuk mengirim pesan yang berisi data biner. Konektor tidak mengurai data biner. Data dialirkan seperti apa adanya.

Untuk memetakan permintaan input untuk respon output, mengganti # wildcard dalam topik pesan dengan ID permintaan arbitrer. Misalnya, jika Anda mempublikasikan pesan ke `kinesisfirehose/message/binary/request123`, properti `id` di objek respons akan ditetapkan ke `request123`.

Jika Anda tidak ingin memetakan permintaan untuk respon, Anda dapat menerbitkan pesan Anda ke `kinesisfirehose/message/binary/`. Pastikan untuk menyertakan garis miring.

Version 1

Filter topik: `kinesisfirehose/message`

Gunakan topik ini untuk mengirim pesan yang berisi data JSON.

Properti pesan

`request`

Data yang akan dikirim ke aliran pengiriman dan aliran pengiriman target, jika berbeda dari aliran default.

Wajib: `true`

Jenis: `object` yang mencakup properti berikut:

`data`

Data untuk mengirim ke aliran pengiriman.

Wajib: `true`

Jenis: `string`

`delivery_stream_arn`

ARN dari target aliran pengiriman Kinesis. Sertakan properti ini untuk menimpa aliran pengiriman default.

Wajib: `false`

Jenis: `string`

Pola yang valid: `arn:aws:firehose:([a-z]{2}-[a-z]+-\d{1}):(\d{12}):deliverystream/([a-zA-Z0-9_\-.]+)$`

`id`

ID arbitrer untuk permintaan. Properti ini digunakan untuk memetakan permintaan input untuk respons output. Ketika ditentukan, `id` properti dalam objek respon diatur ke nilai ini. Jika Anda tidak menggunakan fitur ini, Anda dapat menghilangkan properti ini atau menentukan string kosong.

Wajib: `false`

Jenis: `string`

Pola yang valid: `.*`

Contoh masukan

```
{
     "request": {
        "delivery_stream_arn": "arn:aws:firehose:region:account-
id:deliverystream/stream2-name",
        "data": "Data to send to the delivery stream."
     },
     "id": "request123"
}
```

Filter topik: `kinesisfirehose/message/binary/#`

Gunakan topik ini untuk mengirim pesan yang berisi data biner. Konektor tidak mengurai data biner. Data dialirkan seperti apa adanya.

Untuk memetakan permintaan input untuk respon output, mengganti # wildcard dalam topik pesan dengan ID permintaan arbitrer. Misalnya, jika Anda mempublikasikan pesan ke `kinesisfirehose/message/binary/request123`, properti `id` di objek respons akan ditetapkan ke `request123`.

Jika Anda tidak ingin memetakan permintaan untuk respon, Anda dapat menerbitkan pesan Anda ke `kinesisfirehose/message/binary/`. Pastikan untuk menyertakan garis miring.

Data output

Konektor ini menerbitkan informasi status sebagai data output pada topik MQTT.

Versions 2 - 5

Filter topik dalam langganan

```
kinesisfirehose/message/status
```

Contoh keluaran

Tanggapan berisi status setiap catatan data yang dikirim dalam batch.

```
{
    "response": [
        {
            "ErrorCode": "error",
            "ErrorMessage": "test error",
            "id": "request123",
            "status": "fail"
        },
        {
            "firehose_record_id": "xyz2",
            "id": "request456",
            "status": "success"
        },
        {
            "firehose_record_id": "xyz3",
            "id": "request890",
            "status": "success"
        }
    ]
}
```

> **Note**
>
> Jika konektor mendeteksi kesalahan yang dapat diulang (sebagai contoh, kesalahan koneksi), konektor mengulang lagi publikasinya dalam batch berikutnya. Backoff eksponensial ditangani oleh SDK. AWS Permintaan yang gagal dengan kesalahan retryable ditambahkan kembali ke akhir antrian untuk penerbitan lebih lanjut.

Version 1

Filter topik dalam langganan

```
kinesisfirehose/message/status
```

Contoh keluaran: Sukses

```
{
   "response": {
       "firehose_record_id": "1lxfuuuFomkpJYzt/34ZU/r8JYPf8Wyf7AXqlXm",
       "status": "success"
    },
    "id": "request123"
}
```

Contoh keluaran: Kegagalan

```
{
   "response" : {
       "error": "ResourceNotFoundException",
       "error_message": "An error occurred (ResourceNotFoundException) when
 calling the PutRecord operation: Firehose test1 not found under account
 123456789012.",
       "status": "fail"
   },
   "id": "request123"
}
```

Contoh Penggunaan

Gunakan langkah-langkah tingkat tinggi berikut untuk mengatur contoh fungsi Lambda Python 3.7 yang dapat Anda gunakan untuk mencoba konektor.

> **Note**
>
> - Jika Anda menggunakan waktu aktif Python lainnya, Anda dapat membuat symlink dari Python3.x ke Python 3.7.

- Topik Memulai dengan konektor (konsol) dan Memulai dengan konektor (CLI) berisi langkah-langkah terperinci yang menunjukkan cara mengonfigurasi dan men-deploy contoh konektor Notifikasi Twilio.

1. Pastikan Anda memenuhi persyaratan untuk konektor.

   Untuk persyaratan peran grup, Anda harus mengonfigurasi peran untuk memberikan izin yang diperlukan dan memastikan peran telah ditambahkan ke grup. Untuk informasi lebih lanjut, lihat the section called “Mengelola peran grup (konsol)” atau the section called “Kelola peran grup (CLI)”.
2. Buat dan terbitkan fungsi Lambda yang mengirimkan data input ke konektor.

   Simpan kode contoh sebagai file PY. Unduh dan unzip AWS IoT Greengrass Core SDK for Python. Kemudian, buat paket zip yang berisi file PY dan folder `greengrasssdk` dalam tingkat root. Paket zip ini adalah paket deployment yang Anda unggah ke AWS Lambda.

   Setelah Anda membuat fungsi Lambda Python 3.7, terbitkankan versi fungsi dan buat alias.
3. Konfigurasikan grup Greengrass Anda.

   a. Tambahkan fungsi Lambda dengan aliasnya (direkomendasikan). Konfigurasikan siklus hidup Lambda sebagai berumur panjang (atau `"Pinned": true` dalam CLI).

   b. Tambahkan konektor dan konfigurasikan parameter.

   c. Tambahkan langganan yang mengizinkan konektor untuk menerima data input JSON dan mengirim data output pada filter topik yang didukung.

      - Atur fungsi Lambda sebagai sumber, konektor sebagai target, dan gunakan filter topik input yang mendukung.
      - Atur konektor sebagai sumber, AWS IoT Core sebagai target, dan gunakan filter topik input yang mendukung. Anda menggunakan langganan ini untuk melihat pesan status di AWS IoT konsol.
4. Men-deploy grup.
5. Di AWS IoT konsol, pada halaman Uji, berlangganan topik data keluaran untuk melihat pesan status dari konektor. Contoh fungsi Lambda yang berumur panjang dan mulai mengirim pesan segera setelah grup dalam-deploy.

Setelah selesai pengujian, Anda dapat mengatur siklus hidup Lambda ke sesuai permintaan (atau `"Pinned": false` dalam CLI) dan men-deploy grup. Ini menghentikan fungsi dari mengirim pesan.

Contoh

Contoh fungsi Lambda berikut mengirimkan pesan input ke konektor. Pesan ini berisi data JSON.

```
import greengrasssdk
import time
import json

iot_client = greengrasssdk.client('iot-data')
send_topic = 'kinesisfirehose/message'

def create_request_with_all_fields():
    return  {
        "request": {
            "data": "Message from Firehose Connector Test"
        },
        "id" : "req_123"
    }

def publish_basic_message():
    messageToPublish = create_request_with_all_fields()
    print("Message To Publish: ", messageToPublish)
    iot_client.publish(topic=send_topic,
        payload=json.dumps(messageToPublish))

publish_basic_message()

def lambda_handler(event, context):
    return
```

Lisensi

Konektor Kinesis Firehose mencakup perangkat lunak/lisensi pihak ketiga berikut ini:

- AWS SDK for Python (Boto3)/Lisensi 2.0 Apache
- botocore/Apache License 2.0

- dateutil/PSF License
- docutils/BSD License, Lisensi Publik Umum (GPL) GNU, Lisensi Dasar Perangkat Lunak Python, Domain Publik
- jmespath/MIT License
- s3transfer/Apache License 2.0
- urllib3/Lisensi MIT

Konektor ini dirilis di bawah Perjanjian Lisensi Perangkat lunak core Greengrass.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi konektor.

Versi	Perubahan
5	Ditambahkan parameter `IsolationMode` untuk mengonfigurasi mode kontainerisasi untuk konektor.
4	Memperbarui waktu aktif Lambda untuk Python 3.7, yang mengubah persyaratan waktu aktif.
3	Perbaiki untuk mengurangi logging berlebihan dan perbaikan bug kecil lainnya.
2	Menambahkan dukungan untuk mengirim catatan data batch ke Firehose pada interval tertentu. • Juga memerlukan `firehose:PutRecordBatch` tindakan dalam peran grup. • Parameter `MemorySize` , `DeliveryStreamQueueSize` , dan `PublishInterval` parameter. • Pesan output berisi array respon status untuk catatan data yang diterbitkan.

Versi	Perubahan
1	Pelepasan awal.

Sebuah grup Greengrass dapat berisi hanya satu versi konektor pada suatu waktu. Untuk informasi lebih lanjut tentang pembaruan versi konektor, lihat the section called “Versi upgrade konektor”.

Lihat juga

- Integrasikan dengan layanan dan protokol menggunakan konektor
- the section called “Memulai dengan konektor (konsol)”
- the section called “Memulai dengan konektor (CLI)”
- Apa Amazon Kinesis Data Firehose? dalam Panduan Developer Amazon Kinesis

Konektor Umpan balik ML

> **Warning**
>
> Konektor ini telah pindah kefase kehidupan yang diperpanjang, danAWS IoT Greengrasstidak akan merilis pembaruan yang menyediakan fitur, penyempurnaan pada fitur yang ada, tambalan keamanan, atau perbaikan bug. Untuk informasi selengkapnya, lihat AWS IoT Greengrass Version 1kebijakan pemeliharaan.

Konektor Umpan balik ML memudahkan Anda mengakses data model machine learning (ML) untuk pelatihan ulang dan analisis model. Konektor:

- Unggah data input (sampel) yang digunakan oleh model ML Anda ke Amazon S3. Model input dapat dalam format apa pun, seperti gambar, JSON, atau audio. Setelah sampel diunggah ke cloud, Anda dapat menggunakannya untuk melatih model untuk meningkatkan akurasi dan ketepatan prediksi. Sebagai contoh, Anda dapat menggunakan SageMaker Ground Truth untuk melabeli sampel Anda dan SageMaker untuk melatih kembali model.
- Menerbitkan hasil prediksi dari model sebagai pesan MQTT. Hal ini memungkinkan Anda memantau dan menganalisis kualitas inferensi model Anda secara real time. Anda juga dapat menyimpan hasil prediksi dan menggunakannya untuk menganalisis tren dari waktu ke waktu.
- Menerbitkan metrik tentang sampel unggahan dan data sampel ke Amazon CloudWatch.

Untuk mengonfigurasi konektor ini, Anda menjelaskan dukungan Anda konfigurasi umpan balik dalam format JSON. Sebuah konfigurasi umpan balik mendefinisikan properti seperti tujuan bucket Amazon S3, jenis konten, dan strategi sampling. (Strategi pengambilan sampel digunakan untuk menentukan sampel mana yang akan diunggah.)

Anda dapat menggunakan konektor Umpan balik ML dalam skenario berikut:

- Dengan fungsi Lambda yang ditetapkan pengguna. Fungsi Lambda inferensi lokal Anda menggunakan Machine Learning SDK AWS IoT Greengrass untuk memanggil konektor ini dan lulus dalam konfigurasi umpan balik target, input model, dan output model (hasil prediksi). Sebagai contoh, lihat the section called “Contoh Penggunaan”.
- Dengan konektor Klasifikasi Citra ML (v2). Untuk menggunakan konektor ini dengan konektor Klasifikasi Citra ML, konfigurasikan parameter `MLFeedbackConnectorConfigId` untuk konektor Klasifikasi Citra ML.
- Dengan konektor Deteksi Objek ML. Untuk menggunakan konektor ini dengan konektor Deteksi Objek ML, konfigurasikan parameter `MLFeedbackConnectorConfigId` untuk konektor Deteksi Objek ML.

ARN: `arn:aws:greengrass:region::/connectors/MLFeedback/versions/1`

Persyaratan

Konektor ini memiliki persyaratan sebagai berikut:

- AWS IoT Greengrass perangkat lunak Core v1.9.3 atau yang lebih baru.
- Python versi 3.7 atau 3.8 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.

> **Note**
>
> Untuk menggunakan Python 3.8, jalankan perintah berikut untuk membuat link simbolik dari folder instalasi default Python 3.7 ke binari Python 3.8 yang diinstal.
>
> ```
> sudo ln -s path-to-python-3.8/python3.8 /usr/bin/python3.7
> ```
>
> Ini mengonfigurasi perangkat Anda untuk memenuhi persyaratan Python untuk AWS IoT Greengrass.

- Satu atau lebih bucket Amazon S3. Jumlah bucket yang Anda gunakan tergantung pada strategi pengambilan sampel Anda.
- Peran grup Greengrass dikonfigurasi untuk mengizinkan `s3:PutObject` tindakan pada objek dalam tujuan bucket Amazon S3, seperti yang ditunjukkan dalam contoh kebijakan IAM berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::bucket-name/*"
            ]
        }
    ]
}
```

  Kebijakan harus mencakup semua bucket tujuan sebagai sumber. Anda dapat memberikan akses terperinci atau bersyarat ke sumber daya (sebagai contoh, dengan menggunakan skema penamaan wildcard *).

  Untuk persyaratan peran grup, Anda harus mengonfigurasi peran untuk memberikan izin yang diperlukan dan memastikan peran telah ditambahkan ke grup. Untuk informasi lebih lanjut, lihat the section called “Mengelola peran grup (konsol)” atau the section called “Kelola peran grup (CLI)”.
- Konektor Metrik CloudWatch ditambahkan ke grup Greengrass dan dikonfigurasi. Hal ini diperlukan hanya jika Anda ingin menggunakan fitur pelaporan metrik.
- AWS IoT Greengrass Machine Learning SDK v1.1.0 diperlukan untuk berinteraksi dengan konektor ini.

Parameter

`FeedbackConfigurationMap`

Satu set dari satu atau lebih konfigurasi umpan balik yang konektor dapat gunakan untuk mengunggah sampel ke Amazon S3. Konfigurasi umpan balik mendefinisikan parameter seperti bucket tujuan, jenis konten, dan strategi pengambilan sampel. Ketika konektor ini dipanggil, memanggil fungsi Lambda atau konektor menentukan konfigurasi umpan balik target.

Nama tampilanAWS IoTKonsol: Peta konfigurasi umpan balik

Wajib: `true`

Jenis: Sebuah string JSON terbentuk yang mendefinisikan himpunan konfigurasi umpan balik yang didukung. Sebagai contoh, lihat the section called “Contoh FeedBackConfigurationMap”.

ID objek konfigurasi umpan balik memiliki persyaratan sebagai berikut.

ID:

- Harus unik dalam seluruh objek konfigurasi.
- Harus mulai dengan angka atau huruf kecil. Dapat berisi huruf kecil dan huruf besar, angka, dan tanda hubung.
- Panjangnya harus 2 - 63 karakter.

Wajib: `true`

Jenis: `string`

Pola yang valid: `^[a-zA-Z0-9][a-zA-Z0-9-]{1,62}$`

Contoh:`MyConfig0`,`config-a`,`12id`

Tubuh objek konfigurasi umpan balik berisi properti berikut.

`s3-bucket-name`

Nama bucket Amazon S3 tujuan.

> **Note**
>
> Peran grup harus mengizinkan tindakan `s3:PutObject` pada semua bucket tujuan. Untuk informasi selengkapnya, lihat the section called “Persyaratan”.

Wajib: `true`

Jenis: `string`

Pola yang valid: `^[a-z0-9\.\-]{3,63}$`

`content-type`

Jenis konten sampel untuk diunggah. Semua konten untuk konfigurasi umpan balik individu harus dari jenis yang sama.

Wajib: `true`

Jenis: `string`

Contoh:`image/jpeg,application/json,audio/ogg`

`s3-prefix`

Prefiks kunci yang digunakan untuk sampel yang diunggah. Prefix serupa dengan nama direktori. Itu mengizinkan menyimpan data serupa di bawah direktori yang sama di bucket. Untuk informasi selengkapnya, lihatKunci dan metadata objekdiPanduan Pengguna Amazon Simple Storage Service.

Wajib: `false`

Jenis: `string`

`file-ext`

Ekstensi file yang digunakan untuk sampel yang diunggah. Harus ekstensi file yang valid untuk jenis konten.

Wajib: `false`

Jenis: `string`

Contoh:`jpg,json,ogg`

`sampling-strategy`

Untuk Strategi sampling yang digunakan untuk memfilter sampel yang diunggah. Jika dihilangkan, konektor mencoba untuk mengunggah semua sampel yang diterimanya.

Wajib: `false`

Jenis: Sebuah string JSON terbentuk yang berisi properti berikut.

`strategy-name`

Nama strategi sampling.

Wajib: `true`

Jenis: `string`

Nilai yang valid: `RANDOM_SAMPLING`, `LEAST_CONFIDENCE`, `MARGIN`, atau `ENTROPY`

`rate`

Tingkat untuk strategi sampling acak.

Wajib: `true` jika `strategy-name` adalah `RANDOM_SAMPLING`.

Tipe: `number`

Nilai yang valid: `0.0 - 1.0`

`threshold`

Ambang batas untuk KepercayaanTerkecil, Margin, atau Entropi strategi sampling.

Wajib: `true` jika `strategy-name` adalah `LEAST_CONFIDENCE`, `MARGIN`, atau `ENTROPY`.

Tipe: `number`

Nilai yang valid:

- `0.0 - 1.0` untuk `LEAST_CONFIDENCE` atau `MARGIN` strategi.
- `0.0 - no limit` untuk `ENTROPY` strategi.

`RequestLimit`

Jumlah maksimum permintaan yang dapat diproses konektor pada satu waktu.

Anda dapat menggunakan parameter ini untuk membatasi konsumsi memori dengan membatasi jumlah permintaan yang konektor proses pada ketika yang sama. Permintaan yang melebihi batas ini diabaikan.

Nama tampilanAWS IoTKonsol: Batas permintaan

Wajib: `false`

Jenis: `string`

Nilai yang valid: `0 - 999`

Pola yang valid: `^$|^[0-9]{1,3}$`

Buat Contoh Konektor (AWS CLI)

Perintah CLI berikut membuat `ConnectorDefinition` dengan versi awal yang berisi konektor Umpan balik ML.

```
aws greengrass create-connector-definition --name MyGreengrassConnectors --initial-
version '{
    "Connectors": [
        {
            "Id": "MyMLFeedbackConnector",
            "ConnectorArn": "arn:aws:greengrass:region::/connectors/MLFeedback/
versions/1",
            "Parameters": {
                "FeedbackConfigurationMap": "{  \"RandomSamplingConfiguration\":
 {  \"s3-bucket-name\": \"my-aws-bucket-random-sampling\",  \"content-type\":
 \"image/png\",  \"file-ext\": \"png\",  \"sampling-strategy\": {  \"strategy-name
\": \"RANDOM_SAMPLING\",  \"rate\": 0.5  } },  \"LeastConfidenceConfiguration\": {
  \"s3-bucket-name\": \"my-aws-bucket-least-confidence-sampling\",  \"content-type\":
 \"image/png\",  \"file-ext\": \"png\",  \"sampling-strategy\": {  \"strategy-name\":
 \"LEAST_CONFIDENCE\",  \"threshold\": 0.4  } } }",
                "RequestLimit": "10"
            }
        }
    ]
}'
```

Contoh FeedBackConfigurationMap

Berikut ini adalah nilai contoh diperluas untuk parameter `FeedbackConfigurationMap` ini. Contoh ini mencakup beberapa konfigurasi umpan balik yang menggunakan strategi sampling yang berbeda.

```
{
    "ConfigID1": {
        "s3-bucket-name": "my-aws-bucket-random-sampling",
        "content-type": "image/png",
        "file-ext": "png",
        "sampling-strategy": {
            "strategy-name": "RANDOM_SAMPLING",
```

```
            "rate": 0.5
        }
    },
    "ConfigID2": {
        "s3-bucket-name": "my-aws-bucket-margin-sampling",
        "content-type": "image/png",
        "file-ext": "png",
        "sampling-strategy": {
            "strategy-name": "MARGIN",
            "threshold": 0.4
        }
    },
    "ConfigID3": {
        "s3-bucket-name": "my-aws-bucket-least-confidence-sampling",
        "content-type": "image/png",
        "file-ext": "png",
        "sampling-strategy": {
            "strategy-name": "LEAST_CONFIDENCE",
            "threshold": 0.4
        }
    },
    "ConfigID4": {
        "s3-bucket-name": "my-aws-bucket-entropy-sampling",
        "content-type": "image/png",
        "file-ext": "png",
        "sampling-strategy": {
            "strategy-name": "ENTROPY",
            "threshold": 2
        }
    },
    "ConfigID5": {
        "s3-bucket-name": "my-aws-bucket-no-sampling",
        "s3-prefix": "DeviceA",
        "content-type": "application/json"
    }
}
```

Strategi sampling

Konektor mendukung empat strategi sampling yang menentukan apakah akan mengunggah sampel yang dilewatkan ke konektor. Sampel adalah contoh diskrit data yang menggunakan model untuk prediksi. Anda dapat menggunakan strategi sampling untuk memfilter sampel yang paling mungkin untuk meningkatkan akurasi model.

`RANDOM_SAMPLING`

Mengunggah sampel secara acak berdasarkan tingkat yang disediakan. Mengunggah sampel jika nilai yang dihasilkan secara acak kurang dari tingkat. Semakin tinggi tingkat, semakin banyak sampel yang diunggah.

> **Note**
>
> Strategi ini mengabaikan prediksi model yang disediakan.

`LEAST_CONFIDENCE`

Mengunggah sampel yang probabilitas kepercayaan maksimum jatuh di bawah ambang batas yang disediakan.

Contoh skenario:

Ambang batas: `.6`

Prediksi model: `[.2, .2, .4, .2]`

Keyakinan maksimum probabilitas: `.4`

Hasil:

Gunakan sampel karena probabilitas kepercayaan maksimum (`.4`) <= ambang batas (`.6`).

`MARGIN`

Unggah sampel jika margin antara dua probabilitas kepercayaan jatuh dalam ambang batas yang disediakan. Margin adalah perbedaan antara dua probabilitas teratas.

Contoh skenario:

Ambang batas: `.02`

Prediksi model: `[.3, .35, .34, .01]`

Top dua probabilitas kepercayaan: `[.35, .34]`

Margin: `.01` (`.35 - .34`)

Hasil:

Gunakan sampel karena margin (`.01`) <= ambang batas (`.02`).

`ENTROPY`

Unggah sampel yang entropi lebih besar dari ambang batas yang disediakan. Gunakan model prediksi entropi dinormalisasi.

Contoh skenario:

Ambang batas: `0.75`

Prediksi model: `[.5, .25, .25]`

Entropi untuk prediksi: `1.03972`

Hasil:

Gunakan sampel karena entropi (`1.03972`) > ambang batas (`0.75`).

Data input

Fungsi Lambda yang ditetapkan pengguna menggunakan fungsi `publish` dari klien `feedback` dalam AWS IoT Greengrass Machine Learning SDK untuk memanggil konektor. Sebagai contoh, lihat the section called “Contoh Penggunaan”.

> **Note**
>
> Konektor ini tidak menerima pesan MQTT sebagai data input.

Fungsi `publish` mengambil argumen berikut:

ConfigId

ID dari konfigurasi umpan balik target. Ini harus sesuai dengan ID dari konfigurasi umpan balik yang didefinisikan dalam FeedbackConfigurationMap parameter untuk konektor Umpan balik ML.

Wajib: BETUL

Tipe: string

ModelInput

Data input yang dilewatkan ke model untuk inferensi. Data input ini diunggah menggunakan konfigurasi target kecuali disaring berdasarkan strategi sampling.

Wajib: BETUL

Jenis: byte

ModelPrediction

Hasil prediksi dari model. Jenis hasil dapat berupa kamus atau daftar. Sebagai contoh, hasil prediksi dari konektor Klasifikasi Citra ML adalah daftar probabilitas (seperti `[0.25, 0.60, 0.15]`). Data ini diterbitkan ke topik `/feedback/message/prediction` ini.

Wajib: BETUL

Jenis: kamus atau daftar `float` nilai

Metadata

Pelanggan didefinisikan, aplikasi-spesifik metadata yang terlampir pada sampel yang diunggah dan diterbitkan ke `/feedback/message/prediction` Topik. Konektor juga menyisipkan sebuah `publish-ts` kunci dengan nilai timestamp ke metadata.

Wajib: SALAH

Jenis: kamus

Contoh: `{"some-key": "some value"}`

Data output

Konektor ini menerbitkan data ke tiga topik MQTT:

- Informasi status dari konektor pada `feedback/message/status` topik.
- Hasil prediksi pada `feedback/message/prediction` topik.
- Metrik ditakdirkan untuk CloudWatch pada`cloudwatch/metric/put`topik.

Anda harus mengonfigurasi langganan untuk mengizinkan konektor berkomunikasi pada topik MQTT. Untuk informasi selengkapnya, lihat the section called “Input dan output”.

Filter topik: `feedback/message/status`

Gunakan topik ini untuk memantau status unggah sampel dan batalkan sampel. Konektor menerbitkan topik ini setiap kali menerima permintaan.

Contoh output: Pengunggahan sampel berhasil


```
{
  "response": {
    "status": "success",
    "s3_response": {
      "ResponseMetadata": {
        "HostId": "IOWQ4fDEXAMPLEQM+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK
+Jd1vEXAMPLEa3Km",
        "RetryAttempts": 1,
        "HTTPStatusCode": 200,
        "RequestId": "79104EXAMPLEB723",
        "HTTPHeaders": {
          "content-length": "0",
          "x-amz-id-2":
 "lbbqaDVFOhMlyU3gRvAX1ZIdg8P0WkGkCSSFsYFvSwLZk3j7QZhG5EXAMPLEdd4/pEXAMPLEUqU=",
          "server": "AmazonS3",
          "x-amz-expiration": "expiry-date=\"Wed, 17 Jul 2019 00:00:00 GMT\",
 rule-id=\"OGZjYWY3OTgtYWI2Zi00ZDllLWE4YmQtNzMyYzEXAMPLEoUw\"",
          "x-amz-request-id": "79104EXAMPLEB723",
          "etag": "\"b9c4f172e64458a5fd674EXAMPLE5628\"",
          "date": "Thu, 11 Jul 2019 00:12:50 GMT",
          "x-amz-server-side-encryption": "AES256"
        }
      },
      "bucket": "greengrass-feedback-connector-data-us-west-2",
      "ETag": "\"b9c4f172e64458a5fd674EXAMPLE5628\"",
      "Expiration": "expiry-date=\"Wed, 17 Jul 2019 00:00:00 GMT\", rule-id=
\"OGZjYWY3OTgtYWI2Zi00ZDllLWE4YmQtNzMyYzEXAMPLEoUw\"",
      "key": "s3-key-prefix/UUID.file_ext",
      "ServerSideEncryption": "AES256"
    }
  },
  "id": "5aaa913f-97a3-48ac-5907-18cd96b89eeb"
}
```


Konektor menambahkan `bucket` dan `key` bidang untuk respon dari Amazon S3. Untuk informasi lebih lanjut tentang respons Amazon s3, lihat objek PUT dalam Referensi Amazon Simple Storage Service API.

Contoh output: Sampel dibatalkan karena strategi pengambilan sampel


```
{
```

```
  "response": {
    "status": "sample_dropped_by_strategy"
  },
  "id": "4bf5aeb0-d1e4-4362-5bb4-87c05de78ba3"
}
```

Contoh output: Pengunggahan sampel gagal

Status kegagalan mencakup pesan gagal sebagai `error_message` nilai dan kelas pengecualian sebagai `error` nilai.

```
{
  "response": {
    "status": "fail",
    "error_message": "[RequestId: 4bf5aeb0-d1e4-4362-5bb4-87c05de78ba3] Failed
 to upload model input data due to exception. Model prediction will not be
 published. Exception type: NoSuchBucket, error: An error occurred (NoSuchBucket)
 when calling the PutObject operation: The specified bucket does not exist",
    "error": "NoSuchBucket"
  },
  "id": "4bf5aeb0-d1e4-4362-5bb4-87c05de78ba3"
}
```

Contoh output: Permintaan terhambat karena batas permintaan

```
{
  "response": {
    "status": "fail",
    "error_message": "Request limit has been reached (max request: 10 ). Dropping
 request.",
    "error": "Queue.Full"
  },
  "id": "4bf5aeb0-d1e4-4362-5bb4-87c05de78ba3"
}
```

Filter topik: `feedback/message/prediction`

Gunakan topik ini untuk mendengarkan prediksi berdasarkan data sampel yang diunggah. Hal ini memungkinkan Anda menganalisis kinerja model Anda secara real time. Prediksi model diterbitkan ke topik ini hanya jika data berhasil diunggah ke Amazon S3. Pesan yang diterbitkan pada topik ini dalam format JSON. Mereka berisi link ke objek data yang diunggah, prediksi model, dan metadata termasuk dalam permintaan.

Anda juga dapat menyimpan hasil prediksi dan menggunakannya untuk melaporkan dan menganalisis tren dari waktu ke waktu. Tren dapat memberikan pandangan yang berharga. Sebagai contoh, sebuah tren penurunan akurasi dari waktu ke waktu dapat membantu Anda untuk memutuskan apakah model perlu dilatih ulang.

Contoh output

```
{
  "source-ref": "s3://greengrass-feedback-connector-data-us-west-2/s3-key-prefix/
UUID.file_ext",
  "model-prediction": [
    0.5,
    0.2,
    0.2,
    0.1
  ],
  "config-id": "ConfigID2",
  "metadata": {
    "publish-ts": "2019-07-11 00:12:48.816752"
  }
}
```

> **Tip**
>
> Anda dapat mengonfigurasi konektor IoT Analytics untuk berlangganan topik ini dan mengirim informasi ke AWS IoT Analytics untuk analisis lebih lanjut atau sejarah.

Filter topik: `cloudwatch/metric/put`

Ini adalah topik output yang digunakan untuk menerbitkan metrik ke CloudWatch. Fitur ini mengharuskan Anda menginstal dan mengonfigurasi Konektor Metrik CloudWatch.

Termasuk metrik:

- Jumlah sampel yang diunggah.
- Ukuran sampel yang diunggah.
- Jumlah kesalahan dari unggahan ke Amazon S3.
- Jumlah sampel batal berdasarkan strategi sampling.
- Jumlah permintaan yang ditahan.

Contoh output: Ukuran sampel data (diterbitkan sebelum unggah sebenarnya)

```
{
  "request": {
    "namespace": "GreengrassFeedbackConnector",
    "metricData": {
      "value": 47592,
      "unit": "Bytes",
      "metricName": "SampleSize"
    }
  }
}
```

Contoh output: Pengunggahan sampel berhasil

```
{
  "request": {
    "namespace": "GreengrassFeedbackConnector",
    "metricData": {
      "value": 1,
      "unit": "Count",
      "metricName": "SampleUploadSuccess"
    }
  }
}
```

Contoh output: Pengunggahan sampel berhasil dan hasil prediksi diterbitkan

```
{
  "request": {
    "namespace": "GreengrassFeedbackConnector",
    "metricData": {
      "value": 1,
      "unit": "Count",
      "metricName": "SampleAndPredictionPublished"
    }
  }
}
```

Contoh output: Pengunggahan sampel gagal

```
{
```

```
  "request": {
    "namespace": "GreengrassFeedbackConnector",
    "metricData": {
      "value": 1,
      "unit": "Count",
      "metricName": "SampleUploadFailure"
    }
  }
}
```

Contoh output: Sampel dibatalkan karena strategi pengambilan sampel

```
{
  "request": {
    "namespace": "GreengrassFeedbackConnector",
    "metricData": {
      "value": 1,
      "unit": "Count",
      "metricName": "SampleNotUsed"
    }
  }
}
```

Contoh output: Permintaan terhambat karena batas permintaan

```
{
  "request": {
    "namespace": "GreengrassFeedbackConnector",
    "metricData": {
      "value": 1,
      "unit": "Count",
      "metricName": "ErrorRequestThrottled"
    }
  }
}
```

Contoh Penggunaan

Contoh berikut adalah fungsi Lambda yang ditetapkan pengguna yang menggunakan AWS IoT Greengrass Machine Learning SDK untuk mengirim data ke konektor Umpan balik ML.

> **Note**
>
> Anda dapat mengunduh AWS IoT Greengrass Machine Learning SDK dari AWS IoT Greengrass halaman unduh.

```
import json
import logging
import os
import sys
import greengrass_machine_learning_sdk as ml

client = ml.client('feedback')

try:
    feedback_config_id = os.environ["FEEDBACK_CONFIG_ID"]
    model_input_data_dir = os.environ["MODEL_INPUT_DIR"]
    model_prediction_str = os.environ["MODEL_PREDICTIONS"]
    model_prediction = json.loads(model_prediction_str)
except Exception as e:
    logging.info("Failed to open environment variables. Failed with exception:
{}".format(e))
    sys.exit(1)

try:
    with open(os.path.join(model_input_data_dir, os.listdir(model_input_data_dir)[0]),
 'rb') as f:
        content = f.read()
except Exception as e:
    logging.info("Failed to open model input directory. Failed with exception:
{}".format(e))
    sys.exit(1)

def invoke_feedback_connector():
    logging.info("Invoking feedback connector.")
    try:
        client.publish(
            ConfigId=feedback_config_id,
            ModelInput=content,
            ModelPrediction=model_prediction
        )
    except Exception as e:
```

```
        logging.info("Exception raised when invoking feedback connector:{}".format(e))
        sys.exit(1)

invoke_feedback_connector()

def function_handler(event, context):
    return
```

Lisensi

Konektor Umpan balik ML mencakup perangkat lunak/lisensi pihak ketiga berikut:

- AWS SDK for Python (Boto3)/Lisensi 2.0 Apache
- botocore/Apache License 2.0
- dateutil/PSF License
- docutils/BSD License, Lisensi Publik Umum (GPL) GNU, Lisensi Dasar Perangkat Lunak Python, Domain Publik
- jmespath/MIT License
- s3transfer/Apache License 2.0
- urllib3/Lisensi MIT
- six/MIT

Konektor ini dirilis di bawah Perjanjian Lisensi Perangkat lunak core Greengrass.

Lihat juga

- Integrasikan dengan layanan dan protokol menggunakan konektor
- the section called “Memulai dengan konektor (konsol)”
- the section called “Memulai dengan konektor (CLI)”

Konektor Klasifikasi Citra ML

> **Warning**
>
> Konektor ini telah pindah ke fase umur yang diperpanjang, dan AWS IoT Greengrass tidak akan merilis pembaruan yang menyediakan fitur, penyempurnaan pada fitur yang ada, patch keamanan, atau perbaikan bug. Untuk informasi selengkapnya, lihat AWS IoT Greengrass Version 1kebijakan pemeliharaan.

Konektor Klasifikasi Citra ML menyediakan layanan inferensi machine learning (ML) yang berjalan dalam AWS IoT Greengrass core. Layanan inferensi lokal ini melakukan klasifikasi gambar menggunakan model yang dilatih oleh algoritma klasifikasi SageMaker gambar.

Fungsi Lambda yang ditetapkan pengguna menggunakan Machine Learning SDK AWS IoT Greengrass untuk mengirimkan permintaan inferensi ke layanan inferensi lokal. Layanan ini berjalan inferensi lokal dan mengembalikan probabilitas bahwa gambar input milik kategori tertentu.

AWS IoT Greengrass menyediakan versi berikut dari konektor ini, yang tersedia untuk beberapa platform.

Version 2

Konektor	Deskripsi dan ARN
Klasifikasi Citra ML Aarch64 JTX2	Layanan inferensi klasifikasi citra untuk NVIDIA Jetson TX2. Mendukung akselerasi GPU. ARN: `arn:aws:greengrass:region::/connectors/ImageClassificationAarch64JTX2/versions/2`
Klasifikasi Citra ML x86_64	Layanan inferensi klasifikasi citra untuk platform x86_64. ARN: `arn:aws:greengrass:region::/connectors/Imag`

Konektor	Deskripsi dan ARN
	`eClassificationx86-64/versions/2`
Klasifikasi Citra ML ARMv7	Layanan inferensi klasifikasi citra untuk platform ARMv7. ARN: `arn:aws:greengrass:` *`region`*`::/connectors/ImageClassificationARMv7/versions/2`

Version 1

Konektor	Deskripsi dan ARN
Klasifikasi Citra ML Aarch64 JTX2	Layanan inferensi klasifikasi citra untuk NVIDIA Jetson TX2. Mendukung akselerasi GPU. ARN: `arn:aws:greengrass:` *`region`*`::/connectors/ImageClassificationAarch64JTX2/versions/1`
Klasifikasi Citra ML x86_64	Layanan inferensi klasifikasi citra untuk platform x86_64. ARN: `arn:aws:greengrass:` *`region`*`::/connectors/ImageClassificationx86-64/versions/1`
Klasifikasi Citra ML Armv7	Layanan inferensi klasifikasi citra untuk platform Armv7.

Konektor	Deskripsi dan ARN
	ARN: `arn:aws:greengrass:region::/connectors/ImageClassificationARMv7/versions/1`

Untuk informasi tentang perubahan versi, lihat Changelog.

Persyaratan

Konektor-konektor ini memiliki persyaratan sebagai berikut:

Version 2

- AWS IoT Greengrass perangkat lunak Core v1.9.3 atau yang lebih baru.
- Python versi 3.7 atau 3.8 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.

  > **Note**
  >
  > Untuk menggunakan Python 3.8, jalankan perintah berikut untuk membuat link simbolik dari folder instalasi default Python 3.7 ke binari Python 3.8 yang diinstal.
  >
  > ```
  > sudo ln -s path-to-python-3.8/python3.8 /usr/bin/python3.7
  > ```
  >
  > Ini mengonfigurasi perangkat Anda untuk memenuhi persyaratan Python untuk AWS IoT Greengrass.

- Ketergantungan untuk kerangka Apache MXNet diinstal pada perangkat core. Untuk informasi selengkapnya, lihat the section called “Menginstal dependensi MXNet”.
- Sumber daya ML dalam grup Greengrass yang mereferensikan sumber model. SageMaker Model ini harus dilatih oleh algoritma klasifikasi SageMaker gambar. Untuk informasi selengkapnya, lihat Algoritma klasifikasi gambar di Panduan SageMaker Pengembang Amazon.
- Konektor Umpan balik ML ditambahkan ke grup Greengrass dan dikonfigurasi. Hal ini diperlukan hanya jika Anda ingin menggunakan konektor untuk mengunggah model data input dan menerbitkan prediksi untuk topik MQTT.

- Peran grup Greengrass dikonfigurasi untuk mengizinkan tindakan `sagemaker:DescribeTrainingJob` pada tugas pelatihan target, seperti yang ditunjukkan dalam contoh kebijakan IAM berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sagemaker:DescribeTrainingJob"
            ],
            "Resource": "arn:aws:sagemaker:region:account-id:training-
job:training-job-name"
        }
    ]
}
```

  Untuk persyaratan peran grup, Anda harus mengonfigurasi peran untuk memberikan izin yang diperlukan dan memastikan peran telah ditambahkan ke grup. Untuk informasi lebih lanjut, lihat the section called “Mengelola peran grup (konsol)” atau the section called “Kelola peran grup (CLI)”.

  Anda dapat memberikan akses terperinci atau bersyarat ke sumber daya (sebagai contoh, dengan menggunakan skema penamaan wildcard *). Jika Anda mengubah target pekerjaan pelatihan dalam masa depan, pastikan untuk memperbarui peran grup.

- AWS IoT Greengrass Machine Learning SDK v1.1.0 diperlukan untuk berinteraksi dengan konektor ini.

Version 1

- AWS IoT Greengrass perangkat lunak Core v1.7 atau yang lebih baru.
- Python versi 2.7 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.
- Ketergantungan untuk kerangka Apache MXNet diinstal pada perangkat core. Untuk informasi selengkapnya, lihat the section called “Menginstal dependensi MXNet”.
- Sumber daya ML dalam grup Greengrass yang mereferensikan sumber model. SageMaker Model ini harus dilatih oleh algoritma klasifikasi SageMaker gambar. Untuk informasi selengkapnya, lihat Algoritma klasifikasi gambar di Panduan SageMaker Pengembang Amazon.

- Peran grup Greengrass dikonfigurasi untuk mengizinkan tindakan `sagemaker:DescribeTrainingJob` pada tugas pelatihan target, seperti yang ditunjukkan dalam contoh kebijakan IAM berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sagemaker:DescribeTrainingJob"
            ],
            "Resource": "arn:aws:sagemaker:region:account-id:training-
job:training-job-name"
        }
    ]
}
```

  Untuk persyaratan peran grup, Anda harus mengonfigurasi peran untuk memberikan izin yang diperlukan dan memastikan peran telah ditambahkan ke grup. Untuk informasi lebih lanjut, lihat the section called “Mengelola peran grup (konsol)” atau the section called “Kelola peran grup (CLI)”.

  Anda dapat memberikan akses terperinci atau bersyarat ke sumber daya (sebagai contoh, dengan menggunakan skema penamaan wildcard *). Jika Anda mengubah target pekerjaan pelatihan dalam masa depan, pastikan untuk memperbarui peran grup.

- AWS IoT Greengrass Machine Learning SDK v1.0.0 atau yang lebih baru diperlukan untuk berinteraksi dengan konektor ini.

Parameter Konektor

Konektor-konektor ini menyediakan parameter berikut.

Version 2

`MLModelDestinationPath`

Jalur lokal absolut dari sumber daya ML dalam lingkungan Lambda. Ini adalah jalur tujuan yang ditentukan untuk sumber daya ML.

> **Note**
>
> Jika Anda membuat sumber daya ML dalam konsol, ini adalah jalur lokal.

Nama tampilan pada konsol AWS IoT tersebut: Model jalur tujuan

Wajib: `true`

Jenis: `string`

Pola yang valid: `.+`

`MLModelResourceId`

ID dari sumber daya ML yang mereferensikan model sumber.

Nama tampilan di AWS IoT konsol: sumber daya ARN SageMaker pekerjaan

Wajib: `true`

Jenis: `string`

Pola yang valid: `[a-zA-Z0-9:_-]+`

`MLModelSageMakerJobArn`

ARN dari pekerjaan SageMaker pelatihan yang mewakili sumber SageMaker model. Model harus dilatih oleh algoritma klasifikasi SageMaker gambar.

Nama tampilan di AWS IoT konsol: SageMaker pekerjaan ARN

Wajib: `true`

Jenis: `string`

Pola yang valid: `^arn:aws:sagemaker:[a-zA-Z0-9-]+:[0-9]+:training-job/[a-zA-Z0-9][a-zA-Z0-9-]+$`

`LocalInferenceServiceName`

Nama untuk layanan inferensi lokal. Fungsi Lambda yang ditentukan pengguna memanggil layanan dengan melewatkan nama ke fungsi `invoke_inference_service` dari AWS

IoT Greengrass Machine Learning SDK. Sebagai contoh, lihat the section called “Contoh Penggunaan”.

Nama tampilan pada konsol AWS IoT tersebut: Nama layanan inferensi lokal

Wajib: `true`

Jenis: `string`

Pola yang valid: `[a-zA-Z0-9][a-zA-Z0-9-]{1,62}`

`LocalInferenceServiceTimeoutSeconds`

Jumlah waktu (dalam detik) sebelum permintaan kesimpulan dihentikan. Nilai minimum adalah 1.

Nama tampilan pada konsol AWS IoT tersebut: Timeout (detik)

Wajib: `true`

Jenis: `string`

Pola yang valid: `[1-9][0-9]*`

`LocalInferenceServiceMemoryLimitKB`

Jumlah memori (dalam KB) bahwa layanan memiliki akses ke. Nilai minimum adalah 1.

Nama tampilan pada konsol AWS IoT tersebut: Batas memori (KB)

Wajib: `true`

Jenis: `string`

Pola yang valid: `[1-9][0-9]*`

`GPUAcceleration`

CPU atau GPU (dipercepat) konteks komputasi. Properti ini hanya berlaku untuk konektor Klasifikasi Citra ML Aarch64 JTX2.

Nama tampilan pada konsol AWS IoT tersebut: Akselerasi GPU

Wajib: `true`

Jenis: `string`

Nilai yang valid: `CPU or GPU`

`MLFeedbackConnectorConfigId`

ID dari konfigurasi umpan balik untuk digunakan untuk mengunggah model data input. Ini harus sesuai dengan ID dari konfigurasi umpan balik yang ditetapkan untuk Konektor Umpan balik ML.

Parameter ini diperlukan hanya jika Anda ingin menggunakan konektor Umpan balik ML untuk mengunggah model data input dan menerbitkan prediksi untuk topik MQTT.

Nama tampilan pada konsol AWS IoT tersebut: ID konfigurasi konektor umpan balik ML

Wajib: `false`

Jenis: `string`

Pola yang valid: `^$|^[a-zA-Z0-9][a-zA-Z0-9-]{1,62}$`

Version 1

`MLModelDestinationPath`

Jalur lokal absolut dari sumber daya ML dalam lingkungan Lambda. Ini adalah jalur tujuan yang ditentukan untuk sumber daya ML.

> **Note**
>
> Jika Anda membuat sumber daya ML dalam konsol, ini adalah jalur lokal.

Nama tampilan pada konsol AWS IoT tersebut: Model jalur tujuan

Wajib: `true`

Jenis: `string`

Pola yang valid: `.+`

`MLModelResourceId`

ID dari sumber daya ML yang mereferensikan model sumber.

Nama tampilan di AWS IoT konsol: sumber daya ARN SageMaker pekerjaan

Wajib: `true`

Jenis: `string`

Pola yang valid: `[a-zA-Z0-9:_-]+`

`MLModelSageMakerJobArn`

ARN dari pekerjaan SageMaker pelatihan yang mewakili sumber SageMaker model. Model harus dilatih oleh algoritma klasifikasi SageMaker gambar.

Nama tampilan di AWS IoT konsol: SageMaker pekerjaan ARN

Wajib: `true`

Jenis: `string`

Pola yang valid: `^arn:aws:sagemaker:[a-zA-Z0-9-]+:[0-9]+:training-job/[a-zA-Z0-9][a-zA-Z0-9-]+$`

`LocalInferenceServiceName`

Nama untuk layanan inferensi lokal. Fungsi Lambda yang ditentukan pengguna memanggil layanan dengan melewatkan nama ke fungsi `invoke_inference_service` dari AWS IoT Greengrass Machine Learning SDK. Sebagai contoh, lihat the section called “Contoh Penggunaan”.

Nama tampilan pada konsol AWS IoT tersebut: Nama layanan inferensi lokal

Wajib: `true`

Jenis: `string`

Pola yang valid: `[a-zA-Z0-9][a-zA-Z0-9-]{1,62}`

`LocalInferenceServiceTimeoutSeconds`

Jumlah waktu (dalam detik) sebelum permintaan kesimpulan dihentikan. Nilai minimum adalah 1.

Nama tampilan pada konsol AWS IoT tersebut: Timeout (detik)

Wajib: `true`

Jenis: `string`

Pola yang valid: `[1-9][0-9]*`

`LocalInferenceServiceMemoryLimitKB`

Jumlah memori (dalam KB) bahwa layanan memiliki akses ke. Nilai minimum adalah 1.

Nama tampilan pada konsol AWS IoT tersebut: Batas memori (KB)

Wajib: `true`

Jenis: `string`

Pola yang valid: `[1-9][0-9]*`

`GPUAcceleration`

CPU atau GPU (dipercepat) konteks komputasi. Properti ini hanya berlaku untuk konektor Klasifikasi Citra ML Aarch64 JTX2.

Nama tampilan pada konsol AWS IoT tersebut: Akselerasi GPU

Wajib: `true`

Jenis: `string`

Nilai yang valid: `CPU` or `GPU`

Buat Contoh Konektor (AWS CLI)

Perintah CLI berikut membuat `ConnectorDefinition` dengan versi awal yang berisi konektor Klasifikasi Citra ML.

Contoh: Instans CPU

Contoh ini membuat sebuah instans dari konektor Armv7l Klasifikasi Citra ML.

```
aws greengrass create-connector-definition --name MyGreengrassConnectors --initial-
version '{
    "Connectors": [
        {
```

```
            "Id": "MyImageClassificationConnector",
            "ConnectorArn": "arn:aws:greengrass:region::/connectors/
ImageClassificationARMv7/versions/2",
            "Parameters": {
                "MLModelDestinationPath": "/path-to-model",
                "MLModelResourceId": "my-ml-resource",
                "MLModelSageMakerJobArn": "arn:aws:sagemaker:us-
west-2:123456789012:training-job:MyImageClassifier",
                "LocalInferenceServiceName": "imageClassification",
                "LocalInferenceServiceTimeoutSeconds": "10",
                "LocalInferenceServiceMemoryLimitKB": "500000",
                "MLFeedbackConnectorConfigId": "MyConfig0"
            }
        }
    ]
}'
```

Contoh: Instans GPU

Contoh ini menciptakan sebuah permintaan dari konektor Klasifikasi Citra ML Aarch64 JTX2, yang mendukung akselerasi GPU pada papan NVIDIA Jetson TX2.

```
aws greengrass create-connector-definition --name MyGreengrassConnectors --initial-
version '{
    "Connectors": [
        {
            "Id": "MyImageClassificationConnector",
            "ConnectorArn": "arn:aws:greengrass:region::/connectors/
ImageClassificationAarch64JTX2/versions/2",
            "Parameters": {
                "MLModelDestinationPath": "/path-to-model",
                "MLModelResourceId": "my-ml-resource",
                "MLModelSageMakerJobArn": "arn:aws:sagemaker:us-
west-2:123456789012:training-job:MyImageClassifier",
                "LocalInferenceServiceName": "imageClassification",
                "LocalInferenceServiceTimeoutSeconds": "10",
                "LocalInferenceServiceMemoryLimitKB": "500000",
                "GPUAcceleration": "GPU",
                "MLFeedbackConnectorConfigId": "MyConfig0"
            }
        }
    ]
}'
```

> **Note**
>
> Fungsi Lambda dalam konektor-konektor mempunyai siklus hidup yang berumur panjang ini.

Di konsol AWS IoT Greengrass tersebut, Anda dapat menambahkan konektor dari halaman grup Konektor ini. Untuk informasi lebih lanjut, lihat the section called “Memulai dengan konektor (konsol)”.

Data input

Konektor ini menerima file gambar sebagai input. Input file gambar harus dalam `jpeg` atau `png` format. Untuk informasi selengkapnya, lihat the section called “Contoh Penggunaan”.

Konektor-konektor ini tidak menerima pesan MQTT sebagai data input.

Data output

Konektor ini mengembalikan prediksi diformat untuk objek diidentifikasi dalam gambar input:

```
[0.3,0.1,0.04,...]
```

Prediksi berisi daftar nilai yang sesuai dengan kategori yang digunakan dalam dataset pelatihan selama pelatihan model. Setiap nilai mewakili probabilitas bahwa gambar jatuh di bawah kategori yang sesuai. Kategori dengan probabilitas tertinggi adalah prediksi dominan.

Konektor ini tidak menerbitkan pesan MQTT sebagai data output.

Contoh Penggunaan

Contoh fungsi Lambda berikut menggunakan Machine Learning SDK AWS IoT Greengrass untuk berinteraksi dengan konektor Klasifikasi Citra ML.

> **Note**
>
> Anda dapat mengunduh SDK dari halaman unduh AWS IoT Greengrass Machine Learning SDK ini.

Contoh menginisialisasi klien SDK dan serentak menghubungi fungsi SDK `invoke_inference_service` untuk memanggil layanan inferensi lokal. Ini melewati jenis

algoritme, nama layanan, jenis citra, dan konten citra. Kemudian, contoh mengurai respon layanan untuk mendapatkan hasil probabilitas (prediksi).

Python 3.7

```
import logging
from threading import Timer

import numpy as np

import greengrass_machine_learning_sdk as ml

# We assume the inference input image is provided as a local file
# to this inference client Lambda function.
with open('/test_img/test.jpg', 'rb') as f:
    content = bytearray(f.read())

client = ml.client('inference')

def infer():
    logging.info('invoking Greengrass ML Inference service')

    try:
        resp = client.invoke_inference_service(
            AlgoType='image-classification',
            ServiceName='imageClassification',
            ContentType='image/jpeg',
            Body=content
        )
    except ml.GreengrassInferenceException as e:
        logging.info('inference exception {}("{}")'.format(e.__class__.__name__, e))
        return
    except ml.GreengrassDependencyException as e:
        logging.info('dependency exception {}("{}")'.format(e.__class__.__name__,
 e))
        return

    logging.info('resp: {}'.format(resp))
    predictions = resp['Body'].read().decode("utf-8")
    logging.info('predictions: {}'.format(predictions))

    # The connector output is in the format: [0.3,0.1,0.04,...]
    # Remove the '[' and ']' at the beginning and end.
```

```
    predictions = predictions[1:-1]
    count = len(predictions.split(','))
    predictions_arr = np.fromstring(predictions, count=count, sep=',')

    # Perform business logic that relies on the predictions_arr, which is an array
    # of probabilities.

    # Schedule the infer() function to run again in one second.
    Timer(1, infer).start()
    return

infer()

def function_handler(event, context):
    return
```

Python 2.7

```
import logging
from threading import Timer

import numpy

import greengrass_machine_learning_sdk as gg_ml

# The inference input image.
with open("/test_img/test.jpg", "rb") as f:
    content = f.read()

client = gg_ml.client("inference")


def infer():
    logging.info("Invoking Greengrass ML Inference service")

    try:
        resp = client.invoke_inference_service(
            AlgoType="image-classification",
            ServiceName="imageClassification",
            ContentType="image/jpeg",
            Body=content,
        )
    except gg_ml.GreengrassInferenceException as e:
```

```
        logging.info('Inference exception %s("%s")', e.__class__.__name__, e)
        return
    except gg_ml.GreengrassDependencyException as e:
        logging.info('Dependency exception %s("%s")', e.__class__.__name__, e)
        return

    logging.info("Response: %s", resp)
    predictions = resp["Body"].read()
    logging.info("Predictions: %s", predictions)

    # The connector output is in the format: [0.3,0.1,0.04,...]
    # Remove the '[' and ']' at the beginning and end.
    predictions = predictions[1:-1]
    predictions_arr = numpy.fromstring(predictions, sep=",")
    logging.info("Split into %s predictions.", len(predictions_arr))

    # Perform business logic that relies on predictions_arr, which is an array
    # of probabilities.

    # Schedule the infer() function to run again in one second.
    Timer(1, infer).start()


infer()


# In this example, the required AWS Lambda handler is never called.
def function_handler(event, context):
    return
```

Fungsi `invoke_inference_service` dalam Machine Learning SDK AWS IoT Greengrass menerima argumen berikut.

Pendapat	Deskripsi
`AlgoType`	Nama jenis algoritme yang digunakan untuk inferensi. Saat ini, hanya `image-classification` didukung.

Pendapat	Deskripsi
	Wajib: `true` Jenis: `string` Nilai yang valid: `image-classification`
`ServiceName`	Nama layanan inferensi lokal. Gunakan nama yang Anda tentukan untuk parameter `LocalInferenceServiceName` ketika Anda mengonfigurasi konektor. Wajib: `true` Jenis: `string`
`ContentType`	Jenis mime dari gambar input. Wajib: `true` Jenis: `string` Nilai yang valid: `image/jpeg, image/png`
`Body`	Konten dari file citra input. Wajib: `true` Jenis: `binary`

Menginstal dependensi MXNet pada AWS IoT Greengrass core

Untuk menggunakan konektor Klasifikasi Citra ML, Anda harus menginstal dependensi untuk kerangka Apache MXNet pada perangkat core. Konektor menggunakan kerangka kerja untuk melayani model ML.

> **Note**
>
> Konektor ini dibundel dengan perpustakaan MXNet yang telah dikompilasi sebelumnya, sehingga Anda tidak perlu menginstal kerangka kerja MXNet pada perangkat core.

AWS IoT Greengrass menyediakan skrip untuk menginstal dependensi untuk platform dan perangkat umum berikut (atau untuk digunakan sebagai referensi untuk menginstalnya). Jika Anda menggunakan platform atau perangkat lain, lihat dokumentasi MXNet untuk konfigurasi Anda.

Sebelum menginstal dependensi MXNet, pastikan bahwa sistem perpustakaan yang diperlukan (dengan versi minimum yang ditentukan) ada pada perangkat.

NVIDIA Jetson TX2

1. Instal CUDA Toolkit 9.0 dan cuDNN 7.0. Anda bisa mengikuti petunjuk dalam the section called “Mengatur perangkat lain” dalam tutorial Memulai Dengan.
2. Aktifkan repositori universal sehingga konektor dapat menginstal perangkat lunak terbuka yang dikelola komunitas. Untuk informasi lebih lanjut, lihat Repositori/Ubuntu dalam dokumentasi Ubuntu.

   a. Buka file `/etc/apt/sources.list` ini.

   b. Pastikan bahwa baris berikut tidak berkomentar.

   ```
   deb http://ports.ubuntu.com/ubuntu-ports/ xenial universe
   deb-src http://ports.ubuntu.com/ubuntu-ports/ xenial universe
   deb http://ports.ubuntu.com/ubuntu-ports/ xenial-updates universe
   deb-src http://ports.ubuntu.com/ubuntu-ports/ xenial-updates universe
   ```

3. Simpan salinan skrip penginstalan berikut ke file bernama `nvidiajtx2.sh` pada perangkat core.

   Python 3.7

   ```
   #!/bin/bash
   set -e

   echo "Installing dependencies on the system..."
   echo 'Assuming that universe repos are enabled and checking dependencies...'
   apt-get -y update
   ```

```
apt-get -y dist-upgrade
apt-get install -y liblapack3 libopenblas-dev liblapack-dev libatlas-base-
dev
apt-get install -y python3.7 python3.7-dev

python3.7 -m pip install --upgrade pip
python3.7 -m pip install numpy==1.15.0
python3.7 -m pip install opencv-python || echo 'Error: Unable to install
 OpenCV with pip on this platform. Try building the latest OpenCV from source
 (https://github.com/opencv/opencv).'

echo 'Dependency installation/upgrade complete.'
```

> **Note**
>
> Jika OpenCV tidak berhasil menginstal menggunakan skrip ini, Anda dapat mencoba membangun dari sumber. Untuk informasi lebih lanjut, lihat Instalasi dalam Linux dalam dokumentasi OpenCV, atau lihat sumber daya online lainnya untuk platform Anda.

Python 2.7

```
#!/bin/bash
set -e

echo "Installing dependencies on the system..."
echo 'Assuming that universe repos are enabled and checking dependencies...'
apt-get -y update
apt-get -y dist-upgrade
apt-get install -y liblapack3 libopenblas-dev liblapack-dev libatlas-base-dev
 python-dev

echo 'Install latest pip...'
wget https://bootstrap.pypa.io/get-pip.py
python get-pip.py
rm get-pip.py

pip install numpy==1.15.0 scipy

echo 'Dependency installation/upgrade complete.'
```

4. Dari direktori tempat Anda menyimpan file, jalankan perintah berikut:

```
sudo nvidiajtx2.sh
```

x86_64 (Ubuntu or Amazon Linux)

1. Simpan salinan skrip penginstalan berikut ke file bernama `x86_64.sh` pada perangkat core.

   Python 3.7

```
#!/bin/bash
set -e

echo "Installing dependencies on the system..."

release=$(awk -F= '/^NAME/{print $2}' /etc/os-release)

if [ "$release" == '"Ubuntu"' ]; then
  # Ubuntu. Supports EC2 and DeepLens. DeepLens has all the dependencies
 installed, so
  # this is mostly to prepare dependencies on Ubuntu EC2 instance.
  apt-get -y update
  apt-get -y dist-upgrade

  apt-get install -y libgfortran3 libsm6 libxext6 libxrender1
  apt-get install -y python3.7 python3.7-dev
elif [ "$release" == '"Amazon Linux"' ]; then
  # Amazon Linux. Expect python to be installed already
  yum -y update
  yum -y upgrade

  yum install -y compat-gcc-48-libgfortran libSM libXrender libXext
else
  echo "OS Release not supported: $release"
  exit 1
fi

python3.7 -m pip install --upgrade pip
python3.7 -m pip install numpy==1.15.0
python3.7 -m pip install opencv-python || echo 'Error: Unable to install
 OpenCV with pip on this platform. Try building the latest OpenCV from source
 (https://github.com/opencv/opencv).'
```

```
echo 'Dependency installation/upgrade complete.'
```

> **Note**
>
> Jika OpenCV tidak berhasil menginstal menggunakan skrip ini, Anda dapat mencoba membangun dari sumber. Untuk informasi lebih lanjut, lihat Instalasi dalam Linux dalam dokumentasi OpenCV, atau lihat sumber daya online lainnya untuk platform Anda.

Python 2.7

```
#!/bin/bash
set -e

echo "Installing dependencies on the system..."

release=$(awk -F= '/^NAME/{print $2}' /etc/os-release)

if [ "$release" == '"Ubuntu"' ]; then
  # Ubuntu. Supports EC2 and DeepLens. DeepLens has all the dependencies
 installed, so
  # this is mostly to prepare dependencies on Ubuntu EC2 instance.
  apt-get -y update
  apt-get -y dist-upgrade

  apt-get install -y libgfortran3 libsm6 libxext6 libxrender1 python-dev
 python-pip
elif [ "$release" == '"Amazon Linux"' ]; then
  # Amazon Linux. Expect python to be installed already
  yum -y update
  yum -y upgrade

  yum install -y compat-gcc-48-libgfortran libSM libXrender libXext python-
pip
else
  echo "OS Release not supported: $release"
  exit 1
fi
```

```
pip install numpy==1.15.0 scipy opencv-python

echo 'Dependency installation/upgrade complete.'
```

2. Dari direktori tempat Anda menyimpan file, jalankan perintah berikut:

```
sudo x86_64.sh
```

Armv7 (Raspberry Pi)

1. Simpan salinan skrip penginstalan berikut ke file bernama `armv7l.sh` pada perangkat core.

   Python 3.7

```
#!/bin/bash
set -e

echo "Installing dependencies on the system..."

apt-get update
apt-get -y upgrade

apt-get install -y liblapack3 libopenblas-dev liblapack-dev
apt-get install -y python3.7 python3.7-dev

python3.7 -m pip install --upgrade pip
python3.7 -m pip install numpy==1.15.0
python3.7 -m pip install opencv-python || echo 'Error: Unable to install
 OpenCV with pip on this platform. Try building the latest OpenCV from source
 (https://github.com/opencv/opencv).'

echo 'Dependency installation/upgrade complete.'
```

> **Note**
>
> Jika OpenCV tidak berhasil menginstal menggunakan skrip ini, Anda dapat mencoba membangun dari sumber. Untuk informasi lebih lanjut, lihat Instalasi dalam Linux dalam dokumentasi OpenCV, atau lihat sumber daya online lainnya untuk platform Anda.

Python 2.7

```
#!/bin/bash
set -e

echo "Installing dependencies on the system..."

apt-get update
apt-get -y upgrade

apt-get install -y liblapack3 libopenblas-dev liblapack-dev python-dev

# python-opencv depends on python-numpy. The latest version in the APT
 repository is python-numpy-1.8.2
# This script installs python-numpy first so that python-opencv can be
 installed, and then install the latest
# numpy-1.15.x with pip
apt-get install -y python-numpy python-opencv
dpkg --remove --force-depends python-numpy

echo 'Install latest pip...'
wget https://bootstrap.pypa.io/get-pip.py
python get-pip.py
rm get-pip.py

pip install --upgrade numpy==1.15.0 picamera scipy

echo 'Dependency installation/upgrade complete.'
```

2. Dari direktori tempat Anda menyimpan file, jalankan perintah berikut:

```
sudo bash armv7l.sh
```

> **Note**
>
> Pada Raspberry Pi, menggunakan `pip` untuk menginstal dependensi machine learning adalah operasi intensif memori yang dapat menyebabkan perangkat kehabisan memori dan menjadi tidak responsif. Sebagai solusi, Anda dapat sementara meningkatkan ukuran swap:

Di `/etc/dphys-swapfile`, tingkatkan nilai variabel `CONF_SWAPSIZE` dan kemudian jalankan perintah berikut untuk restart `dphys-swapfile`.

```
/etc/init.d/dphys-swapfile restart
```

Pencatatan dan pemecahan masalah

Bergantung pada pengaturan grup Anda, log peristiwa dan kesalahan ditulis ke CloudWatch Log, sistem file lokal, atau keduanya. Catatan dari konektor ini menggunakan prefix `LocalInferenceServiceName`. Jika konektor berperilaku tidak terduga, periksa log konektor. Ini biasanya berisi informasi debugging yang berguna, seperti dependensi perpustakaan ML yang hilang atau penyebab kegagalan startup konektor.

Jika AWS IoT Greengrass grup dikonfigurasi untuk menulis log lokal, konektor menulis file log ke *`greengrass-root`*`/ggc/var/log/user/`*`region`*`/aws/`. Untuk informasi lebih lanjut tentang Greengrass pencatatan, lihat the section called “Pemantauan dengan AWS IoT Greengrass log”.

Gunakan informasi berikut untuk membantu memecahkan masalah dengan konektor Klasifikasi Citra ML.

Pustaka sistem yang diperlukan

Tab berikut mencantumkan sistem perpustakaan yang diperlukan untuk setiap konektor Klasifikasi Citra ML.

ML Image Classification Aarch64 JTX2

Perpustakaan	Versi Minimum
ld-linux-aarch64.jadi.1	GLIBC_2.17
libc.so.6	GLIBC_2.17
libcublas.so.9.0	tidak berlaku
libcudart.so.9.0	tidak berlaku
libcudnn.so.7	tidak berlaku

Perpustakaan	Versi Minimum
libcufft.so.9.0	tidak berlaku
libcurand.so.9.0	tidak berlaku
libcusolver.so.9.0	tidak berlaku
libgcc_s.so.1	GCC_4.2.0
libgomp.so.1	GOMP_4.0, OMP_1.0
libm.so.6	GLIBC_2.23
libpthread.so.0	GLIBC_2.17
librt.so.1	GLIBC_2.17
libstdc++.so.6	GLIBCXX_3.4.21, CXXABI_1.3.8

ML Image Classification x86_64

Perpustakaan	Versi Minimum
ld-linux-x86-64.jadi.2	GCC_4.0.0
libc.so.6	GLIBC_2.4
libgfortran.so.3	GFORTRAN_1.0
libm.so.6	GLIBC_2.23
libpthread.so.0	GLIBC_2.2.5
librt.so.1	GLIBC_2.2.5
libstdc++.so.6	CXXABI_1.3.8, GLIBCXX_3.4.21

ML Image Classification Armv7

Perpustakaan	Versi Minimum
ld-linux-armhf.jadi.3	GLIBC_2.4
libc.so.6	GLIBC_2.7
libgcc_s.so.1	GCC_4.0.0
libgfortran.so.3	GFORTRAN_1.0
libm.so.6	GLIBC_2.4
libpthread.so.0	GLIBC_2.4
librt.so.1	GLIBC_2.4
libstdc++.so.6	CXXABI_1.3.8, CXXABI_ARM_1.3.3, GLIBCXX_3.4.20

Masalah

Gejala	Solusi
Pada Raspberry Pi, pesan eror berikut dicatat dan Anda tidak menggunakan kamera: `Failed to initialize libdc1394`	Jalankan perintah berikut untuk menonaktifkan driver: `sudo ln /dev/null /dev/raw1394` Operasi ini bersifat sementara dan tautan simbolis akan hilang setelah reboot. Konsultas ikan manual distribusi OS Anda untuk mempelajari cara membuat tautan secara otomatis ketika reboot.

Lisensi

Konektor Klasifikasi Citra ML mencakup perangkat lunak/lisensi pihak ketiga berikut:

- AWS SDK for Python (Boto3)/Lisensi 2.0 Apache
- botocore/Apache License 2.0
- dateutil/PSF License
- docutils/BSD License, Lisensi Publik Umum (GPL) GNU, Lisensi Dasar Perangkat Lunak Python, Domain Publik
- jmespath/MIT License
- s3transfer/Apache License 2.0
- urllib3/Lisensi MIT

- Deep Neural Network Library (DNNL)/Lisensi 2.0 Apache
- OpenMP* Perpustakaan Waktu Aktif/See Lisensi Intel OpenMP Perpustakaan Waktu Aktif.
- mxnet/Lisensi 2.0 Apache
- six/MIT

Lisensi Perpustakaan Waktu Aktif Intel OpenMP. Intel® OpenMP* runtume memiliki lisensi ganda, dengan lisensi komersial (COM) sebagai bagian dari produk Intel® Parallel Studio XE Suite, dan lisensi BSD open source (OSS).

Konektor ini dirilis di bawah Perjanjian Lisensi Perangkat lunak core Greengrass.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi konektor.

Versi	Perubahan
2	Menambahkan `MLFeedbackConnectorConfigId` parameter untuk mendukung penggunaan konektor Umpan Balik ML untuk mengunggah data input model, mempublikasikan prediksi ke topik MQTT, dan menerbitkan metrik ke Amazon. CloudWatch

Versi	Perubahan
1	Pelepasan .

Sebuah grup Greengrass dapat berisi hanya satu versi konektor pada suatu waktu. Untuk informasi lebih lanjut tentang pembaruan versi konektor, lihat the section called “Versi upgrade konektor”.

Lihat juga

- Integrasikan dengan layanan dan protokol menggunakan konektor
- the section called “Memulai dengan konektor (konsol)”
- the section called “Memulai dengan konektor (CLI)”
- Tampilkan inferensi machine learning
- Algoritma klasifikasi gambar di Panduan SageMaker Pengembang Amazon

konektor Deteksi Objek ML

> **Warning**
>
> Konektor ini telah pindah kefase kehidupan diperpanjang, danAWS IoT Greengrasstidak akan merilis pembaruan yang menyediakan fitur, penyempurnaan pada fitur yang ada, tambalan keamanan, atau perbaikan bug. Untuk informasi selengkapnya, lihat AWS IoT Greengrass Version 1kebijakan pemeliharaan.

Konektor Deteksi Objek ML menyediakan layanan inferensi machine learning (ML) yang berjalan dalam core AWS IoT Greengrass ini. Layanan inferensi lokal ini melakukan deteksi objek menggunakan model deteksi objek dikompilasi oleh SageMaker Kompiler deep learning Neo. Dua jenis model deteksi objek didukung: Detector Multibox Tunggal (SSD) dan Anda Hanya Look Once (YOLO) v3. Untuk informasi lebih lanjut, lihat Persyaratan Model Deteksi Objek.

Fungsi Lambda yang ditetapkan pengguna menggunakan Machine Learning SDK AWS IoT Greengrass untuk mengirimkan permintaan inferensi ke layanan inferensi lokal. Layanan ini melakukan inferensi lokal pada gambar input dan mengembalikan daftar prediksi untuk setiap objek yang terdeteksi dalam gambar. Setiap prediksi berisi kategori objek, skor kepercayaan prediksi, dan koordinat piksel yang menentukan kotak batas dalam sekitar objek yang diprediksi.

AWS IoT Greengrass menyediakan konektor Deteksi Objek ML untuk beberapa platform:

Konektor	Deskripsi dan ARN
Deteksi Objek ML Aarch64 JTX2	Layanan inferensi deteksi objek untuk NVIDIA Jetson TX2. Mendukung akselerasi GPU. ARN: `arn:aws:greengrass:` *`region`*`::/connectors/ObjectDetectionAarch64JTX2/versions/1`
Deteksi Objek ML x86_64	Layanan inferensi deteksi objek untuk platform x86_64. ARN: `arn:aws:greengrass:` *`region`*`::/connectors/ObjectDetectionx86-64/versions/1`
Deteksi Objek ML ARMv7	Layanan inferensi deteksi objek untuk platform ARMv7. ARN: `arn:aws:greengrass:` *`region`*`::/connectors/ObjectDetectionARMv7/versions/1`

Persyaratan

Konektor-konektor ini memiliki persyaratan sebagai berikut:

- AWS IoT Greengrass perangkat lunak Core v1.9.3 atau yang lebih baru.
- Python versi 3.7 atau 3.8 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.

> **Note**
>
> Untuk menggunakan Python 3.8, jalankan perintah berikut untuk membuat link simbolik dari folder instalasi default Python 3.7 ke binari Python 3.8 yang diinstal.

```
sudo ln -s path-to-python-3.8/python3.8 /usr/bin/python3.7
```

Ini mengonfigurasi perangkat Anda untuk memenuhi persyaratan Python untuk AWS IoT Greengrass.

- Dependensi untuk SageMaker Waktu aktif deep learning Neo diinstal pada perangkat core. Untuk informasi selengkapnya, lihat the section called “Menginstal dependensi waktu aktif Neo deep learning”.
- Sebuah sumber daya ML dalam grup Greengrass. Sumber daya ML harus mereferensikan bucket Amazon S3 yang berisi model deteksi objek. Untuk informasi lebih lanjut, lihat: Sumber model Amazon S3.

> **Note**
>
> Model harus berupa Shot Multibox Detector or You Only Look Once v3 jenis model deteksi obyek. Ini harus dikompilasi menggunakan SageMaker Kompiler deep learning Neo. Untuk informasi lebih lanjut, lihat Persyaratan Model Deteksi Objek.

- Konektor Umpan balik ML ditambahkan ke grup Greengrass dan dikonfigurasi. Hal ini diperlukan hanya jika Anda ingin menggunakan konektor untuk mengunggah model data input dan menerbitkan prediksi untuk topik MQTT.
- AWS IoT Greengrass Machine Learning SDK v1.1.0 diperlukan untuk berinteraksi dengan konektor ini.

Persyaratan model deteksi objek

Konektor Deteksi Objek ML mendukung Single Shot multibox Detector (SSD) dan You Only Look Once (YOLO) v3 jenis model deteksi objek. Anda dapat menggunakan komponen deteksi objek yang disediakan oleh GluonCV untuk melatih model dengan dataset Anda sendiri. Atau, Anda bisa menggunakan model pra-terlatih dari GluonCV Model Zoo:

- Model SSD pra-terlatih
- Model YOLO v3 pra-terlatih

Model deteksi objek Anda harus dilatih dengan gambar input 512 x 512. Model pra-terlatih dari GluonCV Model Zoo sudah memenuhi persyaratan ini.

Model deteksi objek terlatih harus dikompilasi dengan SageMaker Kompiler deep learning Neo. Ketika mengkompilasi, pastikan perangkat keras target cocok dengan perangkat keras perangkat core Greengrass Anda. Untuk informasi selengkapnya, lihat SageMaker Neodi dalamAmazon SageMaker Panduan Pengembang.

Model dikompilasi harus ditambahkan sebagai sumber daya ML (sumber daya model Amazon S3) ke grup Greengrass yang sama dengan konektor.

Parameter Konektor

Konektor-konektor ini menyediakan parameter berikut.

`MLModelDestinationPath`

Jalur absolut untuk bucket Amazon S3 yang berisi model Neo-compatible ML. Ini adalah jalur tujuan yang ditentukan untuk sumber daya model ML.

Nama tampilan diAWS IoTKonsol: Jalur tujuan model

Wajib: `true`

Jenis: `string`

Pola yang valid: `.+`

`MLModelResourceId`

ID dari sumber daya ML yang mereferensikan model sumber.

Nama tampilan diAWS IoTKonsol: Sumber daya ML-Greengrass

Wajib: `true`

Jenis: `S3MachineLearningModelResource`

Pola yang valid: `^[a-zA-Z0-9:_-]+$`

`LocalInferenceServiceName`

Nama untuk layanan inferensi lokal. Fungsi Lambda yang ditentukan pengguna memanggil layanan dengan melewatkan nama ke fungsi `invoke_inference_service` dari AWS IoT Greengrass Machine Learning SDK. Sebagai contoh, lihat the section called “Contoh Penggunaan”.

Nama tampilan diAWS IoTKonsol: Nama layanan inferensi lokal

Wajib: `true`

Jenis: `string`

Pola yang valid: `^[a-zA-Z0-9][a-zA-Z0-9-]{1,62}$`

`LocalInferenceServiceTimeoutSeconds`

Waktu (dalam detik) sebelum permintaan inferensi dihentikan. Nilai minimum adalah 1. Nilai default adalah 10.

Nama tampilan diAWS IoTKonsol: Timeout (kedua)

Wajib: `true`

Jenis: `string`

Pola yang valid: `^[1-9][0-9]*$`

`LocalInferenceServiceMemoryLimitKB`

Jumlah memori (dalam KB) bahwa layanan memiliki akses ke. Nilai minimum adalah 1.

Nama tampilan diAWS IoTKonsol: Batas memori

Wajib: `true`

Jenis: `string`

Pola yang valid: `^[1-9][0-9]*$`

`GPUAcceleration`

CPU atau GPU (dipercepat) konteks komputasi. Properti ini hanya berlaku untuk konektor Klasifikasi Citra ML Aarch64 JTX2.

Nama tampilan diAWS IoTKonsol: Akselerasi GPU

Wajib: `true`

Jenis: `string`

Nilai yang valid: `CPU` or `GPU`

`MLFeedbackConnectorConfigId`

ID dari konfigurasi umpan balik untuk digunakan untuk mengunggah model data input. Ini harus sesuai dengan ID dari konfigurasi umpan balik yang ditetapkan untuk Konektor Umpan balik ML.

Parameter ini diperlukan hanya jika Anda ingin menggunakan konektor Umpan balik ML untuk mengunggah model data input dan menerbitkan prediksi untuk topik MQTT.

Nama tampilan diAWS IoTKonsol: ID konfigurasi konektor Umpan balik

Wajib: `false`

Jenis: `string`

Pola yang valid: `^$|^[a-zA-Z0-9][a-zA-Z0-9-]{1,62}$`

Buat Contoh Konektor (AWS CLI)

Perintah CLI berikut membuat `ConnectorDefinition` dengan versi awal yang berisi konektor Deteksi Objek ML. Contoh ini menbuat sebuah instance dari konektor Deteksi Objek ML ARMv7l.

```
aws greengrass create-connector-definition --name MyGreengrassConnectors --initial-
version '{
    "Connectors": [
        {
            "Id": "MyObjectDetectionConnector",
            "ConnectorArn": "arn:aws:greengrass:region::/connectors/
ObjectDetectionARMv7/versions/1",
            "Parameters": {
                "MLModelDestinationPath": "/path-to-model",
                "MLModelResourceId": "my-ml-resource",
                "LocalInferenceServiceName": "objectDetection",
                "LocalInferenceServiceTimeoutSeconds": "10",
                "LocalInferenceServiceMemoryLimitKB": "500000",
                "MLFeedbackConnectorConfigId" : "object-detector-random-sampling"
            }
        }
    ]
}'
```

> **Note**
>
> Fungsi Lambda dalam konektor-konektor mempunyai siklus hidup yang berumur panjang ini.

Di konsol AWS IoT Greengrass tersebut, Anda dapat menambahkan konektor dari halaman grup Konektor ini. Untuk informasi lebih lanjut, lihat the section called “Memulai dengan konektor (konsol)”.

Data input

Konektor ini menerima file gambar sebagai input. Input file gambar harus dalam `jpeg` atau `png` format. Untuk informasi selengkapnya, lihat the section called “Contoh Penggunaan”.

Konektor-konektor ini tidak menerima pesan MQTT sebagai data input.

Data output

Konektor ini kembali daftar diformat hasil prediksi untuk objek diidentifikasi dalam gambar input:

```
    {
        "prediction": [
            [
                14,
                0.9384938478469849,
                0.37763649225234985,
                0.5110225081443787,
                0.6697432398796082,
                0.8544386029243469
            ],
            [
                14,
                0.8859519958496094,
                0,
                0.43536216020584106,
                0.3314110040664673,
                0.9538808465003967
            ],
            [
                12,
                0.04128098487854004,
                0.5976729989051819,
                0.5747185945510864,
                0.704264223575592,
                0.857937216758728
            ],
            ...
        ]
    }
```

Setiap prediksi dalam daftar terkandung dalam bucket kotak dan berisi enam nilai:

- Nilai pertama merupakan kategori objek diprediksi untuk objek diidentifikasi. Kategori objek dan nilai yang sesuai ditentukan ketika melatih model machine learning deteksi objek Anda dalam Neo deep learning compiler.
- Nilai kedua adalah skor kepercayaan untuk prediksi kategori objek. Ini merupakan probabilitas bahwa prediksi itu benar.
- Empat nilai terakhir sesuai dengan dimensi piksel yang mewakili kotak batas dalam sekitar objek yang diprediksi dalam gambar.

Konektor ini tidak menerbitkan pesan MQTT sebagai data output.

Contoh Penggunaan

contoh fungsi Lambda berikut menggunakan AWS IoT Greengrass Machine Learning SDK untuk berinteraksi dengan konektor Deteksi Objek ML.

> **Note**
>
> Anda dapat mengunduh SDK dari halaman unduh AWS IoT Greengrass Machine Learning SDK ini.

Contoh menginisialisasi klien SDK dan serentak menghubungi fungsi SDK `invoke_inference_service` untuk memanggil layanan inferensi lokal. Ini melewati jenis algoritme, nama layanan, jenis citra, dan konten citra. Kemudian, contoh mengurai respon layanan untuk mendapatkan hasil probabilitas (prediksi).

```
import logging
from threading import Timer

import numpy as np

import greengrass_machine_learning_sdk as ml

# We assume the inference input image is provided as a local file
# to this inference client Lambda function.
with open('/test_img/test.jpg', 'rb') as f:
    content = bytearray(f.read())

client = ml.client('inference')
```

```
def infer():
    logging.info('invoking Greengrass ML Inference service')

    try:
        resp = client.invoke_inference_service(
            AlgoType='object-detection',
            ServiceName='objectDetection',
            ContentType='image/jpeg',
            Body=content
        )
    except ml.GreengrassInferenceException as e:
        logging.info('inference exception {}("{}")'.format(e.__class__.__name__, e))
        return
    except ml.GreengrassDependencyException as e:
        logging.info('dependency exception {}("{}")'.format(e.__class__.__name__, e))
        return

    logging.info('resp: {}'.format(resp))
    predictions = resp['Body'].read().decode("utf-8")
    logging.info('predictions: {}'.format(predictions))
    predictions = eval(predictions)

    # Perform business logic that relies on the predictions.

    # Schedule the infer() function to run again in ten second.
    Timer(10, infer).start()
    return

infer()

def function_handler(event, context):
    return
```

Fungsi `invoke_inference_service` dalam Machine Learning SDK AWS IoT Greengrass menerima argumen berikut.

Pendapat	Deskripsi
`AlgoType`	Nama jenis algoritme yang digunakan untuk inferensi. Saat ini, hanya `object-detection` didukung.

Pendapat	Deskripsi
	Wajib: `true` Jenis: `string` Nilai yang valid: `object-detection`
`ServiceName`	Nama layanan inferensi lokal. Gunakan nama yang Anda tentukan untuk parameter `LocalInferenceServiceName` ketika Anda mengonfigurasi konektor. Wajib: `true` Jenis: `string`
`ContentType`	Jenis mime dari gambar input. Wajib: `true` Jenis: `string` Nilai yang valid: `image/jpeg, image/png`
`Body`	Konten dari file citra input. Wajib: `true` Jenis: `binary`

Menginstal dependensi waktu aktif Neo deep learning pada AWS IoT Greengrass core

Konektor Deteksi Objek LL dibundel dengan SageMaker Waktu aktif deep learning Neo (DLR). Konektor menggunakan waktu aktif untuk melayani model ML. Untuk menggunakan konektor ini, Anda harus menginstal dependensi untuk DLR pada perangkat core Anda.

Sebelum Anda menginstal dependensi DLR, pastikan bahwa sistem perpustakaan yang diperlukan (dengan versi minimum yang ditentukan) ada pada perangkat.

NVIDIA Jetson TX2

1. Instal CUDA Toolkit 9.0 dan cuDNN 7.0. Anda bisa mengikuti petunjuk dalam the section called “Mengatur perangkat lain” dalam tutorial Memulai Dengan.
2. Aktifkan repositori universal sehingga konektor dapat menginstal perangkat lunak terbuka yang dikelola komunitas. Untuk informasi lebih lanjut, lihat Repositori/Ubuntu dalam dokumentasi Ubuntu.

   a. Buka file `/etc/apt/sources.list` ini.

   b. Pastikan bahwa baris berikut tidak berkomentar.

   ```
   deb http://ports.ubuntu.com/ubuntu-ports/ xenial universe
   deb-src http://ports.ubuntu.com/ubuntu-ports/ xenial universe
   deb http://ports.ubuntu.com/ubuntu-ports/ xenial-updates universe
   deb-src http://ports.ubuntu.com/ubuntu-ports/ xenial-updates universe
   ```

3. Simpan salinan skrip penginstalan berikut ke file bernama `nvidiajtx2.sh` pada perangkat core.

   ```
   #!/bin/bash
   set -e

   echo "Installing dependencies on the system..."
   echo 'Assuming that universe repos are enabled and checking dependencies...'
   apt-get -y update
   apt-get -y dist-upgrade
   apt-get install -y liblapack3 libopenblas-dev liblapack-dev libatlas-base-dev
   apt-get install -y python3.7 python3.7-dev

   python3.7 -m pip install --upgrade pip
   python3.7 -m pip install numpy==1.15.0
   python3.7 -m pip install opencv-python || echo 'Error: Unable to install OpenCV
    with pip on this platform. Try building the latest OpenCV from source (https://
   github.com/opencv/opencv).'

   echo 'Dependency installation/upgrade complete.'
   ```

> **Note**
>
> Jika OpenCV tidak berhasil menginstal menggunakan skrip ini, Anda dapat mencoba membangun dari sumber. Untuk informasi lebih lanjut, lihat Instalasi dalam Linux dalam dokumentasi OpenCV, atau lihat sumber daya online lainnya untuk platform Anda.

4. Dari direktori tempat Anda menyimpan file, jalankan perintah berikut:

```
sudo nvidiajtx2.sh
```

x86_64 (Ubuntu or Amazon Linux)

1. Simpan salinan skrip penginstalan berikut ke file bernama `x86_64.sh` pada perangkat core.

```
#!/bin/bash
set -e

echo "Installing dependencies on the system..."

release=$(awk -F= '/^NAME/{print $2}' /etc/os-release)

if [ "$release" == '"Ubuntu"' ]; then
  # Ubuntu. Supports EC2 and DeepLens. DeepLens has all the dependencies
 installed, so
  # this is mostly to prepare dependencies on Ubuntu EC2 instance.
  apt-get -y update
  apt-get -y dist-upgrade

  apt-get install -y libgfortran3 libsm6 libxext6 libxrender1
  apt-get install -y python3.7 python3.7-dev
elif [ "$release" == '"Amazon Linux"' ]; then
  # Amazon Linux. Expect python to be installed already
  yum -y update
  yum -y upgrade

  yum install -y compat-gcc-48-libgfortran libSM libXrender libXext
else
  echo "OS Release not supported: $release"
  exit 1
```

```
fi

python3.7 -m pip install --upgrade pip
python3.7 -m pip install numpy==1.15.0
python3.7 -m pip install opencv-python || echo 'Error: Unable to install OpenCV
 with pip on this platform. Try building the latest OpenCV from source (https://
github.com/opencv/opencv).'

echo 'Dependency installation/upgrade complete.'
```

> **Note**
>
> Jika OpenCV tidak berhasil menginstal menggunakan skrip ini, Anda dapat mencoba membangun dari sumber. Untuk informasi lebih lanjut, lihat Instalasi dalam Linux dalam dokumentasi OpenCV, atau lihat sumber daya online lainnya untuk platform Anda.

2. Dari direktori tempat Anda menyimpan file, jalankan perintah berikut:

```
sudo x86_64.sh
```

ARMv7 (Raspberry Pi)

1. Simpan salinan skrip penginstalan berikut ke file bernama `armv7l.sh` pada perangkat core.

```
#!/bin/bash
set -e

echo "Installing dependencies on the system..."

apt-get update
apt-get -y upgrade

apt-get install -y liblapack3 libopenblas-dev liblapack-dev
apt-get install -y python3.7 python3.7-dev

python3.7 -m pip install --upgrade pip
python3.7 -m pip install numpy==1.15.0
```

```
python3.7 -m pip install opencv-python || echo 'Error: Unable to install OpenCV
 with pip on this platform. Try building the latest OpenCV from source (https://
github.com/opencv/opencv).'

echo 'Dependency installation/upgrade complete.'
```

> **Note**
>
> Jika OpenCV tidak berhasil menginstal menggunakan skrip ini, Anda dapat mencoba membangun dari sumber. Untuk informasi lebih lanjut, lihat Instalasi dalam Linux dalam dokumentasi OpenCV, atau lihat sumber daya online lainnya untuk platform Anda.

2. Dari direktori tempat Anda menyimpan file, jalankan perintah berikut:

```
sudo bash armv7l.sh
```

> **Note**
>
> Pada Raspberry Pi, menggunakan `pip` untuk menginstal dependensi machine learning adalah operasi intensif memori yang dapat menyebabkan perangkat kehabisan memori dan menjadi tidak responsif. Sebagai solusi, Anda dapat sementara meningkatkan ukuran swap. Di `/etc/dphys-swapfile`, tingkatkan nilai variabel `CONF_SWAPSIZE` dan kemudian jalankan perintah berikut untuk restart `dphys-swapfile`.
>
> ```
> /etc/init.d/dphys-swapfile restart
> ```

Pencatatan dan pemecahan masalah

Tergantung pada pengaturan grup Anda, log peristiwa dan kesalahan ditulis CloudWatch Log, sistem file lokal, atau keduanya. Catatan dari konektor ini menggunakan prefix `LocalInferenceServiceName`. Jika konektor berperilaku tidak terduga, periksa log konektor. Ini biasanya berisi informasi debugging yang berguna, seperti dependensi perpustakaan ML yang hilang atau penyebab kegagalan startup konektor.

Jika AWS IoT Greengrass grup dikonfigurasi untuk menulis log lokal, konektor menulis file log ke *greengrass-root*/ggc/var/log/user/*region*/aws/. Untuk informasi lebih lanjut tentang Greengrass pencatatan, lihat the section called “Pemantauan dengan AWS IoT Greengrass log”.

Gunakan informasi berikut untuk membantu memecahkan masalah dengan Deteksi Objek ML konektor.

Sistem perpustakaan yang diperlukan

Tab berikut mencantumkan perpustakaan sistem yang diperlukan untuk setiap konektor Deteksi Objek ML.

ML Object Detection Aarch64 JTX2

Perpustakaan	Versi Minimum
ld-linux-aarch64.so.1	GLIBC_2.17
libc.so.6	GLIBC_2.17
libcublas.so.9.0	tidak berlaku
libcudart.so.9.0	tidak berlaku
libcudnn.so.7	tidak berlaku
libcufft.so.9.0	tidak berlaku
libcurand.so.9.0	tidak berlaku
libcusolver.so.9.0	tidak berlaku
libgcc_s.so.1	GCC_4.2.0
libgomp.so.1	GOMP_4.0, OMP_1.0
libm.so.6	GLIBC_2.23
libnvinfer.so.4	tidak berlaku
libnvrm_gpu.so	tidak berlaku

Perpustakaan	Versi Minimum
libnvrm.so	tidak berlaku
libnvidia-fatbinaryloader.so.28.2.1	tidak berlaku
libnvos.so	tidak berlaku
libpthread.so.0	GLIBC_2.17
librt.so.1	GLIBC_2.17
libstdc++.so.6	GLIBCXX_3.4.21, CXXABI_1.3.8

ML Object Detection x86_64

Perpustakaan	Versi Minimum
ld-linux-x86-64.so.2	GCC_4.0.0
libc.so.6	GLIBC_2.4
libgfortran.so.3	GFORTRAN_1.0
libm.so.6	GLIBC_2.23
libpthread.so.0	GLIBC_2.2.5
librt.so.1	GLIBC_2.2.5
libstdc++.so.6	CXXABI_1.3.8, GLIBCXX_3.4.21

ML Object Detection ARMv7

Perpustakaan	Versi Minimum
ld-linux-armhf.so.3	GLIBC_2.4
libc.so.6	GLIBC_2.7

Perpustakaan	Versi Minimum
libgcc_s.so.1	GCC_4.0.0
libgfortran.so.3	GFORTRAN_1.0
libm.so.6	GLIBC_2.4
libpthread.so.0	GLIBC_2.4
librt.so.1	GLIBC_2.4
libstdc++.so.6	CXXABI_1.3.8, CXXABI_ARM_1.3.3, GLIBCXX_3.4.20

Masalah

Gejala	Solusi
Pada Raspberry Pi, pesan eror berikut dicatat dan Anda tidak menggunakan kamera: `Failed to initialize libdc1394`	Jalankan perintah berikut untuk menonaktifkan driver: `sudo ln /dev/null /dev/raw1394` Operasi ini bersifat fana. Tautan simbolis menghilang setelah Anda reboot. Konsultasikan manual distribusi OS Anda untuk mempelaja ri cara membuat tautan secara otomatis ketika reboot.

Lisensi

Konektor Deteksi Objek ML termasuk perangkat lunak/lisensi pihak ketiga berikut:

- AWS SDK for Python (Boto3)/Lisensi 2.0 Apache
- botocore/Apache License 2.0
- dateutil/PSF License

- docutils/BSD License, Lisensi Publik Umum (GPL) GNU, Lisensi Dasar Perangkat Lunak Python, Domain Publik
- jmespath/MIT License
- s3transfer/Apache License 2.0
- urllib3/Lisensi MIT

- Deep Learning waktu aktif/Lisensi 2.0 Apache
- six/MIT

Konektor ini dirilis di bawah Perjanjian Lisensi Perangkat lunak core Greengrass.

Lihat juga

- Integrasikan dengan layanan dan protokol menggunakan konektor
- the section called “Memulai dengan konektor (konsol)”
- the section called “Memulai dengan konektor (CLI)”
- Tampilkan inferensi machine learning
- Algoritme deteksi objekdi dalamAmazon SageMaker Panduan Pengembang

Konektor Adaptor Protokol Modbus-RTU

Adaptor Protokol Modbus-RTU konektor polling informasi dari perangkat RTU Modbus yang ada dalam grup AWS IoT Greengrass ini.

Konektor ini menerima parameter untuk permintaan Modbus RTU dari fungsi Lambda yang ditetapkan pengguna. Ini mengirimkan permintaan yang sesuai, dan kemudian menerbitkan respon dari perangkat target sebagai pesan MQTT.

Konektor ini memiliki versi berikut.

Versi	ARN
3	`arn:aws:greengrass:  region::/connectors/ModbusRTUProtocolAdapter/versions/3`

Versi	ARN
2	`arn:aws:greengrass:  region::/connectors/ModbusRTUProtocolAdapter/versions/2`
1	`arn:aws:greengrass:  region::/connectors/ModbusRTUProtocolAdapter/versions/1`

Untuk informasi tentang perubahan versi, lihat Changelog.

Persyaratan

Konektor ini memiliki persyaratan sebagai berikut:

Version 3

- AWS IoT Greengrass perangkat lunak Core v1.9.3 atau yang lebih baru.
- Python versi 3.7 atau 3.8 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.

  > **Note**
  >
  > Untuk menggunakan Python 3.8, jalankan perintah berikut untuk membuat link simbolik dari folder instalasi default Python 3.7 ke binari Python 3.8 yang diinstal.
  >
  > ```
  > sudo ln -s path-to-python-3.8/python3.8 /usr/bin/python3.7
  > ```
  >
  > Ini mengonfigurasi perangkat Anda untuk memenuhi persyaratan Python untuk AWS IoT Greengrass.

- Sebuah hubungan fisik antara core AWS IoT Greengrass dan perangkat Modbus. Core harus terhubung secara fisik ke jaringan Modbus RTU melalui port serial; sebagai contoh port USB.
- Sebuah sumber daya perangkat lokal dalam grup Greengrass yang menunjuk ke port serial Modbus fisik.

- Sebuah fungsi Lambda yang ditetapkan pengguna yang mengirimkan Modbus RTU permintaan parameter untuk konektor ini. Parameter permintaan harus sesuai dengan pola yang diharapkan dan termasuk ID dan alamat perangkat target pada jaringan Modbus RTU. Untuk informasi selengkapnya, lihat the section called “Data input”.

Versions 1 - 2

- AWS IoT Greengrass perangkat lunak Core v1.7 atau yang lebih baru.
- Python versi 2.7 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.
- Sebuah hubungan fisik antara core AWS IoT Greengrass dan perangkat Modbus. Core harus terhubung secara fisik ke jaringan Modbus RTU melalui port serial; sebagai contoh port USB.
- Sebuah sumber daya perangkat lokal dalam grup Greengrass yang menunjuk ke port serial Modbus fisik.
- Sebuah fungsi Lambda yang ditetapkan pengguna yang mengirimkan Modbus RTU permintaan parameter untuk konektor ini. Parameter permintaan harus sesuai dengan pola yang diharapkan dan termasuk ID dan alamat perangkat target pada jaringan Modbus RTU. Untuk informasi selengkapnya, lihat the section called “Data input”.

Parameter Konektor

Konektor ini mendukung parameter berikut:

`ModbusSerialPort-ResourceId`

ID sumber daya perangkat lokal yang mewakili fisik Modbus serial port.

> **Note**
>
> Konektor ini diberikan akses baca-tulis ke sumber daya.

Nama tampilan padaAWS IoTkonsol: Sumber daya port serial Modbus

Wajib: `true`

Jenis: `string`

Pola yang valid: `.+`

`ModbusSerialPort`

Jalur absolut ke port serial Modbus fisik pada perangkat. Ini adalah jalur sumber daya yang ditentukan untuk sumber daya perangkat lokal Modbus.

Nama tampilan padaAWS IoTkonsol: Jalur sumber dari sumber daya port serial Modbus

Wajib: `true`

Jenis: `string`

Pola yang valid: `.+`

Buat Contoh Konektor (AWS CLI)

Perintah CLI berikut membuat `ConnectorDefinition` dengan versi awal yang berisi konektor Adaptor Protokol Modbus-RTU.

```
aws greengrass create-connector-definition --name MyGreengrassConnectors --initial-
version '{
    "Connectors": [
        {
            "Id": "MyModbusRTUProtocolAdapterConnector",
            "ConnectorArn": "arn:aws:greengrass:region::/connectors/
ModbusRTUProtocolAdapter/versions/3",
            "Parameters": {
                "ModbusSerialPort-ResourceId": "MyLocalModbusSerialPort",
                "ModbusSerialPort": "/path-to-port"
            }
        }
    ]
}'
```

> **Note**
>
> Fungsi Lambda dalam konektor mempunyai siklus hidup yang berumur panjang ini.

Di konsol AWS IoT Greengrass tersebut, Anda dapat menambahkan konektor dari halaman grup Konektor ini. Untuk informasi selengkapnya, lihat the section called “Memulai dengan konektor (konsol)”.

> **Note**
>
> Setelah Anda menerapkan konektor Adaptor Protokol Modbus-RTU, Anda dapat menggunakan AWS IoT Things Graph untuk mengatur interaksi antar perangkat dalam grup Anda. Untuk informasi lebih lanjut, lihat Modbus dalam AWS IoT Things Graph Panduan Pengguna.

Data input

Konektor ini menerima permintaan parameter Modbus RTU dari fungsi Lambda yang ditetapkan pengguna pada topik MQTT. Pesan input harus dalam format JSON.

Filter topik dalam langganan

```
modbus/adapter/request
```

Properti pesan

Pesan permintaan bervariasi berdasarkan jenis permintaan Modbus RTU yang diwakilinya. Properti berikut diperlukan untuk semua permintaan:

- Di `request` objek:
  - `operation`. Nama operasi untuk mengeksekusi. Sebagai contoh, tentukan `"operation": "ReadCoilsRequest"` untuk membaca koil. Nilai ini harus string Unicode. Untuk operasi yang didukung, lihat the section called “Permintaan dan tanggapan Modbus RTU”.
  - `device`. Perangkat target permintaan. Nilai ini harus berada dalam antara `0 - 247`.
- Properti `id` ini. ID untuk permintaan. Nilai ini digunakan untuk deduplikasi data dan dikembalikan seperti dalam properti `id` dari semua respon, termasuk respon kesalahan. Nilai ini harus string Unicode.

> **Note**
>
> Jika permintaan Anda mencakup bidang alamat, Anda harus menentukan nilai sebagai bilangan bulat. Sebagai contoh, `"address": 1`.

Parameter lain untuk memasukkan dalam permintaan tergantung pada operasi. Semua parameter permintaan diperlukan kecuali CRC, yang ditangani secara terpisah. Sebagai contoh, lihat the section called “Contoh permintaan dan respons”.

Contoh input: Baca kumparan

```
{
    "request": {
        "operation": "ReadCoilsRequest",
     "device": 1,
     "address": 1,
     "count": 1
    },
    "id": "TestRequest"
}
```

Data output

Konektor ini menerbitkan tanggapan atas permintaan Modbus RTU yang masuk.

Filter topik dalam langganan

`modbus/adapter/response`

Properti pesan

Format pesan respon bervariasi berdasarkan permintaan yang sesuai dan status respon. Sebagai contoh, lihat the section called “Contoh permintaan dan respons”.

> **Note**
>
> Sebuah respon untuk operasi menulis hanyalah gema dari permintaan. Meskipun tidak ada informasi yang berarti dikembalikan untuk menulis tanggapan, itu adalah praktik yang baik untuk memeriksa status respon.

Setiap respon mencakup properti berikut:

- Di `response` objek:
  - `status`. Status permintaan. Status dapat menjadi salah satu dari nilai-nilai yang berikut:
    - `Success`. Permintaan itu valid, dikirim ke jaringan Modbus RTU, dan tanggapan dikembalikan.

    - `Exception`. Permintaan itu valid, dikirim ke jaringan Modbus RTU, dan respon pengecualian dikembalikan. Untuk informasi selengkapnya, lihat the section called “Status respon: Pengecualian”.
    - `No Response`. Permintaan tidak valid, dan konektor menangkap kesalahan sebelum permintaan dikirim melalui jaringan Modbus RTU. Untuk informasi selengkapnya, lihat the section called “Status respon: Tidak ada respon”.
  - `device`. Perangkat yang permintaan dikirim ke.
  - `operation`. Jenis permintaan yang dikirim.
  - `payload`. Isi respon yang dikembalikan. Jika `status` adalah `No Response`, objek ini hanya berisi sebuah `error` properti dengan deskripsi kesalahan (sebagai contoh, `"error": "[Input/Output] No Response received from the remote unit"`).
- Properti `id` ini. ID permintaan, digunakan untuk deduplikasi data.

Contoh output: Sukses

```
{
    "response" : {
        "status" : "success",
        "device": 1,
     "operation": "ReadCoilsRequest",
     "payload": {
         "function_code": 1,
         "bits": [1]
     }
     },
     "id" : "TestRequest"
}
```

Contoh output: Gagal

```
{
    "response" : {
        "status" : "fail",
        "error_message": "Internal Error",
        "error": "Exception",
        "device": 1,
     "operation": "ReadCoilsRequest",
     "payload": {
         "function_code": 129,
         "exception_code": 2
```

```
      }
      },
      "id" : "TestRequest"
}
```

Untuk contoh lainnya, lihat the section called “Contoh permintaan dan respons”.

Permintaan dan tanggapan Modbus RTU

Konektor ini menerima parameter permintaan Modbus RTU sebagai data input dan menerbitkan tanggapan sebagai data output.

Operasi umum berikut ini didukung.

Nama operasi dalam permintaan	Kode fungsi dalam respons
ReadCoilsRequest	01
ReadDiscreteInputsRequest	02
ReadHoldingRegistersRequest	03
ReadInputRegistersRequest	04
WriteSingleCoilRequest	05
WriteSingleRegisterRequest	06
WriteMultipleCoilsRequest	15
WriteMultipleRegistersRequest	16
MaskWriteRegisterRequest	22
ReadWriteMultipleRegistersRequest	23

Contoh permintaan dan respons

Berikut ini adalah contoh permintaan dan tanggapan untuk operasi yang didukung.

Baca Koil

Contoh permintaan:

```
{
    "request": {
        "operation": "ReadCoilsRequest",
     "device": 1,
     "address": 1,
     "count": 1
    },
    "id": "TestRequest"
}
```

Contoh respons:

```
{
    "response": {
        "status": "success",
        "device": 1,
     "operation": "ReadCoilsRequest",
     "payload": {
         "function_code": 1,
         "bits": [1]
     }
     },
     "id" : "TestRequest"
}
```

Baca Input Diskrit

Contoh permintaan:

```
{
    "request": {
        "operation": "ReadDiscreteInputsRequest",
        "device": 1,
        "address": 1,
        "count": 1
    },
    "id": "TestRequest"
}
```

Contoh respons:

```
{
    "response": {
        "status": "success",
        "device": 1,
        "operation": "ReadDiscreteInputsRequest",
        "payload": {
            "function_code": 2,
            "bits": [1]
        }
     },
     "id" : "TestRequest"
}
```

Baca Register Holding

Contoh permintaan:

```
{
    "request": {
        "operation": "ReadHoldingRegistersRequest",
     "device": 1,
     "address": 1,
     "count": 1
    },
    "id": "TestRequest"
}
```

Contoh respons:

```
{
    "response": {
        "status": "success",
        "device": 1,
     "operation": "ReadHoldingRegistersRequest",
     "payload": {
         "function_code": 3,
            "registers": [20,30]
     }
     },
     "id" : "TestRequest"
```

```
}
```

Baca Register Input

Contoh permintaan:

```
{
    "request": {
        "operation": "ReadInputRegistersRequest",
     "device": 1,
     "address": 1,
     "value": 1
    },
    "id": "TestRequest"
}
```

Tulis Kumparan Tunggal

Contoh permintaan:

```
{
    "request": {
        "operation": "WriteSingleCoilRequest",
     "device": 1,
     "address": 1,
     "value": 1
    },
    "id": "TestRequest"
}
```

Contoh respons:

```
{
    "response": {
        "status": "success",
        "device": 1,
     "operation": "WriteSingleCoilRequest",
     "payload": {
         "function_code": 5,
         "address": 1,
         "value": true
     }
     },
```

```
     "id" : "TestRequest"
```

Menulis Register Tunggal

Contoh permintaan:

```
{
    "request": {
        "operation": "WriteSingleRegisterRequest",
     "device": 1,
     "address": 1,
     "value": 1
    },
    "id": "TestRequest"
}
```

Tulis Banyak Koil

Contoh permintaan:

```
{
    "request": {
        "operation": "WriteMultipleCoilsRequest",
     "device": 1,
     "address": 1,
     "values": [1,0,0,1]
    },
    "id": "TestRequest"
}
```

Contoh respons:

```
{
    "response": {
        "status": "success",
        "device": 1,
     "operation": "WriteMultipleCoilsRequest",
     "payload": {
         "function_code": 15,
         "address": 1,
         "count": 4
     }
     },
```

```
     "id" : "TestRequest"
}
```

Tulis Banyak Register

Contoh permintaan:

```
{
    "request": {
        "operation": "WriteMultipleRegistersRequest",
     "device": 1,
     "address": 1,
     "values": [20,30,10]
    },
    "id": "TestRequest"
}
```

Contoh respons:

```
{
    "response": {
        "status": "success",
        "device": 1,
     "operation": "WriteMultipleRegistersRequest",
     "payload": {
         "function_code": 23,
         "address": 1,
         "count": 3
     }
     },
     "id" : "TestRequest"
}
```

Topeng Register Menulis

Contoh permintaan:

```
{
    "request": {
        "operation": "MaskWriteRegisterRequest",
     "device": 1,
     "address": 1,
        "and_mask": 175,
```

```
        "or_mask": 1
    },
    "id": "TestRequest"
}
```

Contoh respons:

```
{
    "response": {
        "status": "success",
        "device": 1,
     "operation": "MaskWriteRegisterRequest",
     "payload": {
         "function_code": 22,
            "and_mask": 0,
            "or_mask": 8
     }
     },
     "id" : "TestRequest"
}
```

Baca Tulis Banyak Register

Contoh permintaan:

```
{
    "request": {
        "operation": "ReadWriteMultipleRegistersRequest",
     "device": 1,
     "read_address": 1,
        "read_count": 2,
        "write_address": 3,
        "write_registers": [20,30,40]
    },
    "id": "TestRequest"
}
```

Contoh respons:

```
{
    "response": {
        "status": "success",
        "device": 1,
```

```
    "operation": "ReadWriteMultipleRegistersRequest",
    "payload": {
        "function_code": 23,
        "registers": [10,20,10,20]
    }
    },
    "id" : "TestRequest"
}
```

> **Note**
>
> Register kembali dalam respon ini adalah register yang dibaca dari.

Status respon: Pengecualian

Pengecualian dapat terjadi bila format permintaan valid, tetapi permintaan tersebut tidak berhasil diselesaikan. Dalam kasus ini, respons berisi informasi berikut:

- `status` ditetapkan ke `Exception`.
- `function_code` sama dengan kode fungsi permintaan + 128.
- `exception_code` berisi kode pengecualian. Untuk informasi selengkapnya, lihat kode pengecualian Modbus.

Contoh:

```
{
    "response" : {
        "status" : "fail",
        "error_message": "Internal Error",
        "error": "Exception",
        "device": 1,
     "operation": "ReadCoilsRequest",
     "payload": {
         "function_code": 129,
         "exception_code": 2
     }
     },
     "id" : "TestRequest"
}
```

Status respon: Tidak ada respon

Konektor ini melakukan pemeriksaan validasi pada permintaan Modbus. Sebagai contoh, ia memeriksa format yang tidak sah dan kolom yang hilang. Jika validasi gagal, konektor tidak akan mengirim permintaan. Sebaliknya, ia mengembalikan respon yang berisi informasi berikut:

- Ini `status` diatur ke `No Response`.
- Ini `error` berisi alasan untuk kesalahan.
- Ini `error_message` berisi pesan kesalahan.

Contoh:

```
{
    "response" : {
        "status" : "fail",
        "error_message": "Invalid address field. Expected <type 'int'>, got <type
 'str'>",
        "error": "No Response",
        "device": 1,
     "operation": "ReadCoilsRequest",
     "payload": {
         "error": "Invalid address field. Expected <type 'int'>, got <type 'str'>"
     }
     },
     "id" : "TestRequest"
}
```

Jika permintaan menargetkan perangkat yang tidak ada atau jika jaringan Modbus RTU tidak berfungsi, Anda mungkin mendapatkan `ModbusIOException`, yang menggunakan format No Response.

```
{
    "response" : {
        "status" : "fail",
        "error_message": "[Input/Output] No Response received from the remote unit",
        "error": "No Response",
        "device": 1,
     "operation": "ReadCoilsRequest",
     "payload": {
         "error": "[Input/Output] No Response received from the remote unit"
     }
```

```
    },
    "id" : "TestRequest"
}
```

Contoh Penggunaan

Gunakan langkah-langkah tingkat tinggi berikut untuk mengatur contoh fungsi Lambda Python 3.7 yang dapat Anda gunakan untuk mencoba konektor.

> **Note**
>
> - Jika Anda menggunakan waktu aktif Python lainnya, Anda dapat membuat symlink dari Python3.x ke Python 3.7.
> - Topik Memulai dengan konektor (konsol) dan Memulai dengan konektor (CLI) berisi langkah-langkah terperinci yang menunjukkan cara mengonfigurasi dan men-deploy contoh konektor Notifikasi Twilio.

1. Pastikan Anda memenuhi persyaratan untuk konektor.
2. Buat dan terbitkan fungsi Lambda yang mengirimkan data input ke konektor.

   Simpan kode contoh sebagai file PY. Unduh dan unzip AWS IoT Greengrass Core SDK for Python. Kemudian, buat paket zip yang berisi file PY dan folder `greengrasssdk` dalam tingkat root. Paket zip ini adalah paket deployment yang Anda unggah ke AWS Lambda.

   Setelah Anda membuat fungsi Lambda Python 3.7, terbitkankan versi fungsi dan buat alias.
3. Konfigurasikan grup Greengrass Anda.

   a. Tambahkan fungsi Lambda dengan aliasnya (direkomendasikan). Konfigurasikan siklus hidup Lambda sebagai berumur panjang (atau `"Pinned": true` dalam CLI).

   b. Tambahkan sumber daya perangkat lokal yang diperlukan dan berikan akses baca/tulis ke fungsi Lambda.

   c. Tambahkan konektor dan konfigurasikan parameter.

   d. Tambahkan langganan yang mengizinkan konektor untuk menerima data input dan mengirim data output pada filter topik yang didukung.

      - Atur fungsi Lambda sebagai sumber, konektor sebagai target, dan gunakan filter topik input yang mendukung.

- Atur konektor sebagai sumber, AWS IoT Core sebagai target, dan gunakan filter topik input yang mendukung. Anda menggunakan langganan ini untuk melihat pesan status dalam konsol AWS IoT tersebut.

4. Men-deploy grup.
5. Di konsol AWS IoT tersebut, pada halaman Tes ini, berlangganan ke topik data output untuk melihat pesan status dari konektor. Contoh fungsi Lambda yang berumur panjang dan mulai mengirim pesan segera setelah grup dalam-deploy.

   Setelah selesai pengujian, Anda dapat mengatur siklus hidup Lambda ke sesuai permintaan (atau `"Pinned": false` dalam CLI) dan men-deploy grup. Ini menghentikan fungsi dari mengirim pesan.

Contoh

Contoh fungsi Lambda berikut mengirimkan pesan input ke konektor.

```
import greengrasssdk
import json

TOPIC_REQUEST = 'modbus/adapter/request'

# Creating a greengrass core sdk client
iot_client = greengrasssdk.client('iot-data')

def create_read_coils_request():
 request = {
  "request": {
   "operation": "ReadCoilsRequest",
   "device": 1,
   "address": 1,
   "count": 1
  },
  "id": "TestRequest"
 }
 return request

def publish_basic_request():
 iot_client.publish(payload=json.dumps(create_read_coils_request()),
 topic=TOPIC_REQUEST)

publish_basic_request()
```

```
def lambda_handler(event, context):
 return
```

Lisensi

Konektor Adaptor Protokol Modbus-RTU mencakup perangkat lunak/lisensi pihak ketiga berikut:

- pymodbus/BSD
- pyserial/BSD

Konektor ini dirilis di bawah Perjanjian Lisensi Perangkat lunak core Greengrass.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi konektor.

Versi	Perubahan
3	Memperbarui waktu aktif Lambda untuk Python 3.7, yang mengubah persyaratan waktu aktif.
2	Tingkatkan konektor ARN untuk Wilayah AWS dukungan. Pencatatan kesalahan yang ditingkatkan.
1	Pelepasan .

Sebuah grup Greengrass dapat berisi hanya satu versi konektor pada suatu waktu. Untuk informasi lebih lanjut tentang pembaruan versi konektor, lihat the section called “Versi upgrade konektor”.

Lihat juga

- Integrasikan dengan layanan dan protokol menggunakan konektor
- the section called “Memulai dengan konektor (konsol)”
- the section called “Memulai dengan konektor (CLI)”

Konektor Adaptor Protokol Modbus-TCP

Konektor Adaptor Protokol Modbus-TCP mengumpulkan data dari perangkat lokal melalui protokol ModBustCP dan menerbitkannya ke aliran `StreamManager` yang dipilih.

Anda juga dapat menggunakan konektor ini dengan IoT SiteWise konektor dan IoT Anda SiteWise gateway. Gateway Anda harus menyediakan konfigurasi untuk konektor. Untuk informasi selengkapnya, lihatMengkonfigurasi sumber Modbus TCPdalam IoT SiteWise panduan pengguna.

> **Note**
>
> Konektor ini berjalan dalam mode isolasi Tanpa kontainer ini, sehingga Anda dapat men-deploy ke grup AWS IoT Greengrass yang berjalan pada kontainer Docker.

Konektor ini memiliki versi berikut.

Versi	ARN
3	`arn:aws:greengrass:  region::/connectors/ModbusTCPConnector/versions/3`
2	`arn:aws:greengrass:  region::/connectors/ModbusTCPConnector/versions/2`
1	`arn:aws:greengrass:  region::/connectors/ModbusTCPConnector/versions/1`

Untuk informasi tentang perubahan versi, lihat Changelog.

Persyaratan

Konektor ini memiliki persyaratan sebagai berikut:

Version 1 - 3

- AWS IoT Greengrass perangkat lunak Core v1.10.2 atau yang lebih baru.
- Pengelola streaming diaktifkan pada AWS IoT Greengrass grup.
- Java 8 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan `PATH` ini.

> **Note**
>
> Konektor ini hanya tersedia dalam wilayah berikut:
>
> - ap-southeast-1
> - ap-southeast-2
> - eu-central-1
> - eu-west-1
> - us-east-1
> - us-west-2
> - cn-north-1

Parameter Konektor

Konektor ini mendukung parameter berikut:

`LocalStoragePath`

Direktori diAWS IoT Greengrasshost bahwa IoT SiteWise konektor dapat menulis data persisten untuk. Direktori default adalah `/var/sitewise`.

Nama tampilan dalamAWS IoTKonsol: Jalur penyimpanan lokal

Wajib: `false`

Jenis: `string`

Pola yang valid: `^\s*$|\/.`

`MaximumBufferSize`

Ukuran maksimum dalam GB untuk IoT SiteWise penggunaan disk. Ukuran default-nya adalah 10GB.

Nama tampilan dalamAWS IoTKonsol: Ukuran buffer disk maksimum

Wajib: `false`

Jenis: `string`

Pola yang valid: `^\s*$|[0-9]+`

`CapabilityConfiguration`

Atur konfigurasi kolektor Modbus TCP yang konektor mengumpulkan data dari dan terhubung ke.

Nama tampilan dalamAWS IoTKonsol: CapabilityConfiguration

Wajib: `false`

Jenis: Sebuah string JSON terbentuk yang mendefinisikan himpunan konfigurasi umpan balik yang didukung.

Berikut ini adalah contoh dari `CapabilityConfiguration`:

```
{
    "sources": [
        {
            "type": "ModBusTCPSource",
            "name": "SourceName1",
            "measurementDataStreamPrefix": "SourceName1_Prefix",
            "destination": {
                "type": "StreamManager",
                "streamName": "SiteWise_Stream_1",
                "streamBufferSize": 8
            },
            "endpoint": {
                "ipAddress": "127.0.0.1",
                "port": 8081,
                "unitId": 1
            },
            "propertyGroups": [
```

```
                {
                    "name": "GroupName",
                    "tagPathDefinitions": [
                        {
                            "type": "ModBusTCPAddress",
                            "tag": "TT-001",
                            "address": "30001",
                            "size": 2,
                            "srcDataType": "float",
                            "transformation": "byteWordSwap",
                            "dstDataType": "double"
                        }
                    ],
                    "scanMode": {
                        "type": "POLL",
                        "rate": 100
                    }
                }
            ]
        }
    ]
}
```

Buat Contoh Konektor (AWS CLI)

Perintah CLI berikut membuat `ConnectorDefinition` dengan versi awal yang berisi konektor Adaptor Protokol Modbus-TCP.

```
aws greengrass create-connector-definition --name MyGreengrassConnectors --initial-
version '
{
    "Connectors": [
        {
            "Id": "MyModbusTCPConnector",
            "ConnectorArn": "arn:aws:greengrass:region::/connectors/ModbusTCP/
versions/3",
            "Parameters": {
                "capability_configuration": "{\"version\":1,\"namespace\":
\"iotsitewise:modbuscollector:1\",\"configuration\":\"{\"sources\":[{\"type
\":\"ModBusTCPSource\",\"name\":\"SourceName1\",\"measurementDataStreamPrefix
\":\"\",\"endpoint\":{\"ipAddress\":\"127.0.0.1\",\"port\":8081,\"unitId\":1},
\"propertyGroups\":[{\"name\":\"PropertyGroupName\",\"tagPathDefinitions\":[{\"type
```

```
\":\"ModBusTCPAddress\",\"tag\":\"TT-001\",\"address\":\"30001\",\"size\":2,
\"srcDataType\":\"hexdump\",\"transformation\":\"noSwap\",\"dstDataType\":\"string
\"}],\"scanMode\":{\"rate\":200,\"type\":\"POLL\"}}],\"destination\":{\"type\":
\"StreamManager\",\"streamName\":\"SiteWise_Stream\",\"streamBufferSize\":10},
\"minimumInterRequestDuration\":200}]}\"}"
            }
        }
    ]
}'
```

> **Note**
>
> Fungsi Lambda dalam konektor mempunyai siklus hidup yang berumur panjang ini.

Data input

Konektor ini tidak menerima pesan MQTT sebagai data input.

Data output

Konektor ini menerbitkan data ke `StreamManager`. Anda harus mengonfigurasi aliran pesan tujuan. Pesan output dari struktur berikut:

```
{
    "alias": "string",
    "messages": [
        {
            "name": "string",
            "value": boolean|double|integer|string,
            "timestamp": number,
            "quality": "string"
        }
    ]
}
```

Lisensi

Konektor Adaptor Protokol Modbus-TCP termasuk perangkat lunak/lisensi pihak ketiga berikut:

- Modbus Petri Digital

Konektor ini dirilis di bawah Perjanjian Lisensi Perangkat lunak core Greengrass.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi konektor.

Versi	Perubahan	Tanggal
3 (direkomendasikan)	Versi ini berisi perbaikan bug.	22 Desember 2021
2	Ditambahkan dukungan untuk string sumber dikodekan ASCII, UTF8, dan ISO8859.	24 Mei 2021
1	Pelepasan .	15 Desember 2020

Sebuah grup Greengrass dapat berisi hanya satu versi konektor pada suatu waktu. Untuk informasi lebih lanjut tentang pembaruan versi konektor, lihat the section called “Versi upgrade konektor”.

Lihat juga

- Integrasikan dengan layanan dan protokol menggunakan konektor
- the section called “Memulai dengan konektor (konsol)”
- the section called “Memulai dengan konektor (CLI)”

Konektor Raspberry Pi GPIO

> **Warning**
>
> Konektor ini telah pindah kefase kehidupan yang diperpanjang, danAWS IoT Greengrasstidak akan merilis pembaruan yang menyediakan fitur, penyempurnaan untuk fitur yang ada, patch keamanan, atau perbaikan bug. Untuk informasi selengkapnya, lihat AWS IoT Greengrass Version 1kebijakan pemeliharaan.

konektor Raspberry pi GPIO mengontrol general-purpose input/output (GPIO) pin pada perangkat Raspberry Pi core.

Konektor ini polling input pin pada interval tertentu dan menerbitkan perubahan keadaan untuk topik MQTT. Hal ini juga menerima membaca dan menulis permintaan sebagai pesan MQTT dari fungsi Lambda yang ditetapkan pengguna. Permintaan tulis digunakan untuk mengatur pin tegangan tinggi atau rendah.

Konektor menyediakan parameter yang Anda gunakan untuk menetapkan pin input dan output. Perilaku ini dikonfigurasi sebelum deployment grup. Hal ini tidak dapat diubah pada ketika waktu aktif.

- Pin input dapat digunakan untuk menerima data dari perangkat periferal.
- Pin output dapat digunakan untuk mengontrol periferal atau mengirim data ke periferal.

Anda dapat menggunakan konektor ini untuk banyak skenario, seperti:

- Mengontrol lampu LED hijau, kuning, dan merah untuk lampu lalu lintas.
- Mengontrol kipas (melekat pada relay listrik) berdasarkan data dari sensor kelembaban.
- Mengingatkan karyawan dalam toko ritel ketika pelanggan menekan tombol.
- Menggunakan lampu pintar untuk mengontrol perangkat IoT lainnya.

> **Note**
>
> Konektor ini tidak sesuai untuk aplikasi yang mempunyai persyaratan secara langsung. Acara dengan durasi pendek mungkin tidak terjawab.

Konektor ini memiliki versi berikut.

Versi	ARN
3	`arn:aws:greengrass:  region::/connectors/RaspberryPiGPIO/versions/3`
2	`arn:aws:greengrass:  region::/connectors/RaspberryPiGPIO/versions/2`

Versi	ARN
1	`arn:aws:greengrass:  region::/connectors/RaspberryPiGPIO/versions/1`

Untuk informasi tentang perubahan versi, lihat Changelog.

Persyaratan

Konektor ini memiliki persyaratan sebagai berikut:

Version 3

- AWS IoT Greengrass perangkat lunak Core v1.9.3 atau yang lebih baru.
- Python versi 3.7 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.
- Raspberry Pi 4 Model B, atau Raspberry Pi 3 Model B/B+. Anda harus tahu urutan pin Raspberry Pi Anda. Untuk informasi selengkapnya, lihat the section called “Urutan Pin GPIO”.
- Sebuah sumber daya perangkat lokal dalam grup Greengrass yang menunjuk ke `/dev/gpiomem` dalam Raspberry Pi. Jika Anda membuat sumber daya dalam konsol tersebut, Anda harus memilih opsi Secara otomatis menambahkan izin grup OS dari grup Linux yang memiliki sumber daya ini. Di API, atur `GroupOwnerSetting.AutoAddGroupOwner` properti untuk `true`.
- Modul RPI.GPIO diinstal pada Raspberry Pi. Di Raspbian, modul ini dipasang secara default. Anda dapat menggunakan perintah berikut untuk menginstal ulang:

  ```
  sudo pip install RPi.GPIO
  ```

Versions 1 - 2

- AWS IoT Greengrass perangkat lunak Core v1.7 atau yang lebih baru.
- Python versi 2.7 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.
- Raspberry Pi 4 Model B, atau Raspberry Pi 3 Model B/B+. Anda harus tahu urutan pin Raspberry Pi Anda. Untuk informasi selengkapnya, lihat the section called “Urutan Pin GPIO”.

- Sebuah sumber daya perangkat lokal dalam grup Greengrass yang menunjuk ke `/dev/gpiomem` dalam Raspberry Pi. Jika Anda membuat sumber daya dalam konsol tersebut, Anda harus memilih opsi Secara otomatis menambahkan izin grup OS dari grup Linux yang memiliki sumber daya ini. Di API, atur `GroupOwnerSetting.AutoAddGroupOwner` properti untuk `true`.
- Modul RPI.GPIO diinstal pada Raspberry Pi. Di Raspbian, modul ini dipasang secara default. Anda dapat menggunakan perintah berikut untuk menginstal ulang:

  ```
  sudo pip install RPi.GPIO
  ```

Urutan Pin GPIO

Konektor Raspberry Pi GPIO mereferensikan GPIO pin oleh skema penomoran sistem yang mendasari System on Chip (SoC), bukan oleh tata letak fisik pin GPIO. Pemesanan fisik pin mungkin bervariasi dalam versi Raspberry Pi. Untuk informasi lebih lanjut, lihat GPIO dalam dokumentasi Raspberry Pi.

Konektor tidak dapat memvalidasi bahwa pin input dan output Anda mengonfigurasi peta dengan benar untuk perangkat keras yang mendasari Raspberry Pi Anda. Jika konfigurasi pin tidak valid, konektor mengembalikan kesalahan waktu aktif ketika mencoba untuk memulai pada perangkat. Untuk mengatasi masalah ini, konfigurasikan konektor dan kemudian terapkan lagi.

> **Note**
>
> Pastikan bahwa periferal untuk pin GPIO terhubung dengan baik untuk mencegah kerusakan komponen.

Parameter Konektor

Konektor ini menyediakan parameter berikut:

`InputGpios`

Daftar jumlah pin GPIO dipisahkan dengan koma untuk mengonfigurasi sebagai input. Opsional menambahkan `U` untuk mengatur resistor pull-up pin, atau `D` untuk mengatur resistor pull-down. Contoh:`"5,6U,7D"`.

Nama tampilan padaAWS IoTkonsol: Pin GPIO

Diperlukan: `false`. Anda harus menentukan pin input, pin output, atau keduanya.

Tipe: `string`

Pola yang valid: `^$|^[0-9]+[UD]?(,[0-9]+[UD]?)*$`

`InputPollPeriod`

Interval (dalam milidetik) antara setiap operasi polling suara, yang memeriksa pin input GPIO untuk perubahan keadaan. Nilai minimum adalah 1.

Nilai ini tergantung pada skenario Anda dan jenis perangkat yang disurvei. Sebagai contoh, nilai `50` harus cukup cepat untuk mendeteksi tombol tekan.

Nama tampilan padaAWS IoTkonsol: Masukan periode pemungutan suara GPIO

Wajib: `false`

Jenis: `string`

Pola yang valid: `^$|^[1-9][0-9]*$`

`OutputGpios`

Daftar pin GPIO yang dipisahkan dengan koma untuk mengonfigurasi sebagai output. Opsional menambahkan `H` untuk menetapkan keadaan tinggi (1), atau `L` untuk mengatur keadaan rendah (0). Contoh:`"8H,9,27L"`.

Nama tampilan padaAWS IoTkonsol: Pin GPIO

Diperlukan: `false`. Anda harus menentukan pin input, pin output, atau keduanya.

Tipe: `string`

Pola yang valid: `^$|^[0-9]+[HL]?(,[0-9]+[HL]?)*$`

`GpioMem-ResourceId`

ID sumber daya perangkat lokal yang mewakili `/dev/gpiomem`.

> **Note**
>
> Konektor ini diberikan akses baca-tulis ke sumber daya.

Nama tampilan padaAWS IoTkonsol: Sumber daya untuk perangkat /gpiomem

Wajib: `true`

Jenis: `string`

Pola yang valid: `.+`

Buat Contoh Konektor (AWS CLI)

Perintah CLI berikut membuat `ConnectorDefinition` dengan versi awal yang berisi konektor Raspberry Pi GPIO.

```
aws greengrass create-connector-definition --name MyGreengrassConnectors --initial-
version '{
    "Connectors": [
        {
            "Id": "MyRaspberryPiGPIOConnector",
            "ConnectorArn": "arn:aws:greengrass:region::/connectors/RaspberryPiGPIO/
versions/3",
            "Parameters": {
                "GpioMem-ResourceId": "my-gpio-resource",
                "InputGpios": "5,6U,7D",
                "InputPollPeriod": 50,
                "OutputGpios": "8H,9,27L"
            }
        }
    ]
}'
```

> **Note**
>
> Fungsi Lambda dalam konektor mempunyai siklus hidup yang berumur panjang ini.

Di konsol AWS IoT Greengrass tersebut, Anda dapat menambahkan konektor dari halaman grup Konektor ini. Untuk informasi lebih lanjut, lihat the section called “Memulai dengan konektor (konsol)”.

Data input

Konektor ini menerima permintaan membaca atau menulis untuk pin GPIO pada dua topik MQTT.

- Baca permintaan pada topik `gpio/+/+/read` ini.

- Tulis permintaan pada `gpio/+/+/write` Topik.

Untuk menerbitkan topik ini, ganti + wildcard dengan core hal nama dan nomor pin target, masing-masing. Misalnya:

```
gpio/core-thing-name/gpio-number/read
```

> **Note**
>
> Ketika ini, ketika Anda membuat langganan yang menggunakan konektor Raspberry Pi GPIO, Anda harus menentukan nilai untuk setidaknya salah satu dari + wildcard dalam topik.

Filter topik: `gpio/+/+/read`

Gunakan topik ini untuk mengarahkan konektor untuk membaca keadaan pin GPIO yang ditentukan dalam topik.

Konektor menerbitkan respons terhadap topik output yang sesuai (sebagai contoh, `gpio/core-thing-name/gpio-number/state`).

Properti pesan

Tidak ada. Pesan yang dikirim ke topik ini diabaikan.

Filter topik: `gpio/+/+/write`

Gunakan topik ini untuk mengirim permintaan menulis ke pin GPIO. Ini mengarahkan konektor untuk mengatur pin GPIO yang ditentukan dalam topik untuk tegangan rendah atau tinggi.

- `0` atur pin ke tegangan rendah.
- `1` atur pin ke tegangan tinggi.

Konektor menerbitkan respon ke output yang sesuai `/state` topik (sebagai contoh, `gpio/core-thing-name/gpio-number/state`).

Properti pesan

Nilai `0` atau `1`, sebagai integer atau string.

Contoh input

```
0
```

Data output

Konektor ini menerbitkan data ke dua topik:

- Perubahan tinggi atau rendah pada `gpio/+/+/state` topik.
- Kesalahan pada topik `gpio/+/error` ini.

Filter topik: `gpio/+/+/state`

Gunakan topik ini untuk mendengarkan perubahan status pada pin input dan respon untuk permintaan baca. Konektor mengembalikan string `"0"` jika pin dalam keadaan rendah, atau `"1"` jika dalam keadaan tinggi.

Ketika menerbitkan topik ini, konektor menggantikan + wildcard dengan core nama sesuatu dan target pin, masing-masing. Misalnya:

```
gpio/core-thing-name/gpio-number/state
```

> **Note**
>
> Ketika ini, ketika Anda membuat langganan yang menggunakan konektor Raspberry Pi GPIO, Anda harus menentukan nilai untuk setidaknya salah satu dari + wildcard dalam topik.

Contoh keluaran

```
0
```

Filter topik: `gpio/+/error`

Gunakan topik ini untuk mendengarkan kesalahan. Konektor menerbitkan topik ini sebagai hasil dari permintaan yang tidak valid (sebagai contoh, ketika perubahan keadaan diminta pada pin input).

Ketika menerbitkan topik ini, konektor menggantikan + wildcard dengan core nama sesuatu.

Contoh keluaran

```
{
```

```
  "topic": "gpio/my-core-thing/22/write",
  "error": "Invalid GPIO operation",
  "long_description": "GPIO 22 is configured as an INPUT GPIO. Write operations
are not permitted."
}
```

Contoh Penggunaan

Gunakan langkah-langkah tingkat tinggi berikut untuk mengatur contoh fungsi Lambda Python 3.7 yang dapat Anda gunakan untuk mencoba konektor.

> **Note**
>
> - Jika Anda menggunakan waktu aktif Python lainnya, Anda dapat membuat symlink dari Python3.x ke Python 3.7.
> - Topik Memulai dengan konektor (konsol) dan Memulai dengan konektor (CLI) berisi langkah-langkah terperinci yang menunjukkan cara mengonfigurasi dan men-deploy contoh konektor Notifikasi Twilio.

1. Pastikan Anda memenuhi persyaratan untuk konektor.
2. Buat dan terbitkan fungsi Lambda yang mengirimkan data input ke konektor.

   Simpan kode contoh sebagai file PY. Unduh dan unzip AWS IoT Greengrass Core SDK for Python. Kemudian, buat paket zip yang berisi file PY dan folder `greengrasssdk` dalam tingkat root. Paket zip ini adalah paket deployment yang Anda unggah ke AWS Lambda.

   Setelah Anda membuat fungsi Lambda Python 3.7, terbitkankan versi fungsi dan buat alias.
3. Konfigurasikan grup Greengrass Anda.

   a. Tambahkan fungsi Lambda dengan aliasnya (direkomendasikan). Konfigurasikan siklus hidup Lambda sebagai berumur panjang (atau `"Pinned": true` dalam CLI).

   b. Tambahkan sumber daya perangkat lokal yang diperlukan dan berikan akses baca/tulis ke fungsi Lambda.

   c. Tambahkan konektor dan konfigurasikan parameter.

   d. Tambahkan langganan yang mengizinkan konektor untuk menerima data input dan mengirim data output pada filter topik yang didukung.

- Atur fungsi Lambda sebagai sumber, konektor sebagai target, dan gunakan filter topik input yang mendukung.
- Atur konektor sebagai sumber, AWS IoT Core sebagai target, dan gunakan filter topik input yang mendukung. Anda menggunakan langganan ini untuk melihat pesan status dalam konsol AWS IoT tersebut.

4. Men-deploy grup.
5. Di konsol AWS IoT tersebut, pada halaman Tes ini, berlangganan ke topik data output untuk melihat pesan status dari konektor. Contoh fungsi Lambda yang berumur panjang dan mulai mengirim pesan segera setelah grup dalam-deploy.

   Setelah selesai pengujian, Anda dapat mengatur siklus hidup Lambda ke sesuai permintaan (atau `"Pinned": false` dalam CLI) dan men-deploy grup. Ini menghentikan fungsi dari mengirim pesan.

Contoh

Contoh fungsi Lambda berikut mengirimkan pesan input ke konektor. Contoh ini mengirimkan permintaan membaca untuk satu set input pin GPIO. Ini menunjukkan cara membangun topik menggunakan core nama sesuatu dan nomor pin.

```
import greengrasssdk
import json
import os

iot_client = greengrasssdk.client('iot-data')
INPUT_GPIOS = [6, 17, 22]

thingName = os.environ['AWS_IOT_THING_NAME']

def get_read_topic(gpio_num):
    return '/'.join(['gpio', thingName, str(gpio_num), 'read'])

def get_write_topic(gpio_num):
    return '/'.join(['gpio', thingName, str(gpio_num), 'write'])

def send_message_to_connector(topic, message=''):
    iot_client.publish(topic=topic, payload=str(message))

def set_gpio_state(gpio, state):
```

```
    send_message_to_connector(get_write_topic(gpio), str(state))

def read_gpio_state(gpio):
    send_message_to_connector(get_read_topic(gpio))

def publish_basic_message():
    for i in INPUT_GPIOS:
     read_gpio_state(i)

publish_basic_message()

def lambda_handler(event, context):
    return
```

Lisensi

Raspberry Pi GPIO; konektor termasuk perangkat lunak/lisensi pihak ketiga berikut:

- RPI.GPIO/MIT

Konektor ini dirilis di bawah Perjanjian Lisensi Perangkat lunak core Greengrass.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi konektor.

Versi	Perubahan
3	Memperbarui waktu aktif Lambda untuk Python 3.7, yang mengubah persyaratan waktu aktif.
2	Tingkatkan konektor ARN untuk Wilayah AWS dukungan.
1	Pelepasan .

Sebuah grup Greengrass dapat berisi hanya satu versi konektor pada suatu waktu. Untuk informasi lebih lanjut tentang pembaruan versi konektor, lihat the section called “Versi upgrade konektor”.

Lihat juga

- Integrasikan dengan layanan dan protokol menggunakan konektor
- the section called “Memulai dengan konektor (konsol)”
- the section called “Memulai dengan konektor (CLI)”
- GPIO dalam dokumentasi Raspberry Pi

Konektor Aliran Serial

> **Warning**
>
> Konektor ini telah pindah kefase kehidupan, danAWS IoT Greengrasstidak akan merilis pembaruan yang menyediakan fitur, penyempurnaan untuk fitur yang ada, patch keamanan, atau perbaikan bug. Untuk informasi selengkapnya, lihat AWS IoT Greengrass Version 1kebijakan pemeliharaan.

Konektor Aliran Serial membaca dan menulis ke port serial pada perangkat AWS IoT Greengrass core.

Konektor ini mendukung dua mode operasi:

- Read-On-Demand. Menerima membaca dan menulis permintaan pada topik MQTT dan menerbitkan respon dari operasi membaca atau status operasi menulis.
- Polling-Read. Membaca dari port serial secara berkala. Mode ini juga mendukung permintaan Read-On-Demand.

> **Note**
>
> Membaca permintaan terbatas untuk maksimum membaca panjang 63994 byte. Menulis permintaan terbatas untuk panjang data maksimum 128000 byte.

Konektor ini memiliki versi berikut.

Versi	ARN
3	`arn:aws:greengrass:  region::/connectors/SerialStream/versions/3`
2	`arn:aws:greengrass:  region::/connectors/SerialStream/versions/2`
1	`arn:aws:greengrass:  region::/connectors/SerialStream/versions/1`

Untuk informasi tentang perubahan versi, lihat Changelog.

Persyaratan

Konektor ini memiliki persyaratan sebagai berikut:

Version 3

- AWS IoT Greengrass perangkat lunak Core v1.9.3 atau yang lebih baru.
- Python versi 3.7 atau 3.8 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.

  > **Note**
  >
  > Untuk menggunakan Python 3.8, jalankan perintah berikut untuk membuat link simbolik dari folder instalasi default Python 3.7 ke binari Python 3.8 yang diinstal.
  >
  > ```
  > sudo ln -s path-to-python-3.8/python3.8 /usr/bin/python3.7
  > ```
  >
  > Ini mengonfigurasi perangkat Anda untuk memenuhi persyaratan Python untuk AWS IoT Greengrass.

- Sebuah sumber daya perangkat lokal dalam grup Greengrass yang menunjuk ke port serial target.

> **Note**
>
> Sebelum Anda men-deploy konektor ini, kami rekomendasikan agar Anda mengatur port serial dan memverifikasi bahwa Anda dapat membaca dan menulis untuk itu.

Versions 1 - 2

- AWS IoT Greengrass perangkat lunak Core v1.7 atau yang lebih baru.
- Python versi 2.7 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.
- Sebuah sumber daya perangkat lokal dalam grup Greengrass yang menunjuk ke port serial target.

> **Note**
>
> Sebelum Anda men-deploy konektor ini, kami rekomendasikan agar Anda mengatur port serial dan memverifikasi bahwa Anda dapat membaca dan menulis untuk itu.

Parameter Konektor

Konektor ini menyediakan parameter berikut:

`BaudRate`

Tingkat baud dari koneksi serial.

Nama tampilan diAWS IoTKonsol: Tingkat baud

Wajib: `true`

Jenis: `string`

Nilai yang valid: `110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 56000, 57600, 115200, 230400`

Pola yang valid: `^110$|^300$|^600$|^1200$|^2400$|^4800$|^9600$|^14400$|^19200$|^28800$|^38400$|^56000$|^57600$|^115200$|^230400$`

`Timeout`

Timeout (dalam detik) untuk operasi membaca.

Nama tampilan diAWS IoTKonsol: Waktu habis

Wajib: `true`

Jenis: `string`

Nilai yang valid: `1 - 59`

Pola yang valid: `^([1-9]|[1-5][0-9])$`

`SerialPort`

Jalur absolut ke port serial fisik pada perangkat. Ini adalah jalur sumber yang ditentukan untuk sumber daya perangkat lokal.

Nama tampilan diAWS IoTKonsol: Port Serial

Wajib: `true`

Jenis: `string`

Pola yang valid: `[/a-zA-Z0-9_-]+`

`SerialPort-ResourceId`

ID sumber daya perangkat lokal yang mewakili port serial fisik.

> **Note**
>
> Konektor ini diberikan akses baca-tulis ke sumber daya.

Nama tampilan diAWS IoTKonsol: Sumber daya port Serial

Wajib: `true`

Jenis: `string`

Pola yang valid: `[a-zA-Z0-9_-]+`

`PollingRead`

Mengatur modus baca: Polling-Read-Read-On-Demand.

- Untuk mode Polling-Read, tentukan `true`. Di mode ini, properti `PollingInterval`, `PollingReadType`, dan `PollingReadLength` diperlukan.
- Untuk mode Read-On-Demand, tentukan `false`. Di mode ini, nilai jenis dan panjang ditentukan dalam permintaan baca.

Nama tampilan diAWS IoTKonsol: Mode baca

Wajib: `true`

Jenis: `string`

Nilai yang valid: `true, false`

Pola yang valid: `^([Tt][Rr][Uu][Ee]|[Ff][Aa][Ll][Ss][Ee])$`

`PollingReadLength`

Panjang data (dalam byte) untuk membaca dalam setiap operasi polling membaca. Ini hanya berlaku ketika menggunakan mode Polling-Read.

Nama tampilan diAWS IoTKonsol: Panjang baca polling

Diperlukan: `false`. Properti ini diperlukan saat`PollingRead`adadalah`true`.

Tipe: `string`

Pola yang valid: `^(|[1-9][0-9]{0,3}|[1-5][0-9]{4}|6[0-2][0-9]{3}|63[0-8][0-9]{2}|639[0-8][0-9]|6399[0-4])$`

`PollingReadInterval`

Interval (dalam detik) dalam mana polling baca berlangsung. Ini hanya berlaku ketika menggunakan mode Polling-Read.

Nama tampilan diAWS IoTKonsol: Interval baca

Diperlukan: `false`. Properti ini diperlukan saat`PollingRead`adadalah`true`.

Tipe: `string`

Nilai yang valid: 1 - 999

Pola yang valid: `^(|[1-9]|[1-9][0-9]|[1-9][0-9][0-9])$`

`PollingReadType`

Jenis data yang thread polling membaca. Ini hanya berlaku ketika menggunakan mode Polling-Read.

Nama tampilan diAWS IoTKonsol: Jenis pemungutan suara

Diperlukan: `false`. Properti ini diperlukan saat`PollingRead`adalah`true`.

Tipe: `string`

Nilai yang valid: `ascii, hex`

Pola yang valid: `^(|[Aa][Ss][Cc][Ii][Ii]|[Hh][Ee][Xx])$`

`RtsCts`

Mengindikasikan apakah akan mengaktifkan kontrol aliran RTS/CTS. Nilai default-nya adalah `false`. Untuk informasi lebih lanjut, lihat RTS, CTS, dan RTR.

Nama tampilan diAWS IoTKonsol: Kontrol aliran RTS/CTS

Wajib: `false`

Jenis: `string`

Nilai yang valid: `true, false`

Pola yang valid: `^(|[Tt][Rr][Uu][Ee]|[Ff][Aa][Ll][Ss][Ee])$`

`XonXoff`

Mengindikasikan apakah akan mengaktifkan kontrol aliran perangkat lunak. Nilai default-nya adalah `false`. Untuk informasi lebih lanjut, lihat Kontrol aliran perangkat lunak.

Nama tampilan diAWS IoTKonsol: Kontrol aliran perangkat lunak

Wajib: `false`

Jenis: `string`

Nilai yang valid: `true, false`

Pola yang valid: `^(|[Tt][Rr][Uu][Ee]|[Ff][Aa][Ll][Ss][Ee])$`

`Parity`

Paritas dari port serial. Nilai default-nya adalah `N`. Untuk informasi lebih lanjut, lihat Paritas.

Nama tampilan diAWS IoTKonsol: Paritas port Serial

Wajib: `false`

Jenis: `string`

Nilai yang valid: `N, E, O, S, M`

Pola yang valid: `^(|[NEOSMneosm])$`

Buat Contoh Konektor (AWS CLI)

Perintah CLI berikut membuat `ConnectorDefinition` dengan versi awal yang berisi konektor Aliran Serial. Ini mengonfigurasi konektor untuk mode Polling-Read.

```
aws greengrass create-connector-definition --name MyGreengrassConnectors --initial-
version '{
    "Connectors": [
        {
            "Id": "MySerialStreamConnector",
            "ConnectorArn": "arn:aws:greengrass:region::/connectors/SerialStream/
versions/3",
            "Parameters": {
                "BaudRate" : "9600",
                "Timeout" : "25",
                "SerialPort" : "/dev/serial1",
                "SerialPort-ResourceId" : "my-serial-port-resource",
                "PollingRead" : "true",
                "PollingReadLength" : "30",
                "PollingReadInterval" : "30",
                "PollingReadType" : "hex"
            }
        }
    ]
```

```
}'
```

Di konsol AWS IoT Greengrass tersebut, Anda dapat menambahkan konektor dari halaman grup Konektor ini. Untuk informasi lebih lanjut, lihat the section called “Memulai dengan konektor (konsol)”.

Data input

Konektor ini menerima membaca atau menulis permintaan untuk port serial pada dua topik MQTT. Pesan input harus dalam format JSON.

- Baca permintaan pada topik `serial/+/read/#` ini.
- Menulis permintaan pada `serial/+/write/#` topik.

Untuk menerbitkan topik ini, ganti + wildcard dengan core nama sesuatu dan # wildcard dengan jalan ke port serial. Misalnya:

```
serial/core-thing-name/read/dev/serial-port
```

Filter topik: `serial/+/read/#`

Gunakan topik ini untuk mengirim permintaan baca sesuai permintaan ke pin serial. Membaca permintaan terbatas untuk maksimum membaca panjang 63994 byte.

Properti pesan

`readLength`

Panjang data untuk dibaca dari port serial.

Wajib: `true`

Jenis: `string`

Pola yang valid: `^[1-9][0-9]*$`

`type`

Jenis data untuk dibaca.

Wajib: `true`

Jenis: `string`

Nilai yang valid: `ascii, hex`

Pola yang valid: `(?i)^(ascii|hex)$`

`id`

ID arbitrer untuk permintaan. Properti ini digunakan untuk memetakan permintaan input untuk respons output.

Wajib: `false`

Jenis: `string`

Pola yang valid: `.+`

Contoh input

```
{
    "readLength": "30",
    "type": "ascii",
    "id": "abc123"
}
```

Filter topik: `serial/+/write/#`

Gunakan topik ini untuk mengirim permintaan tulis ke pin serial. Menulis permintaan terbatas untuk panjang data maksimum 128000 byte.

Properti pesan

`data`

String untuk menulis ke port serial.

Wajib: `true`

Jenis: `string`

Pola yang valid: `^[1-9][0-9]*$`

`type`

Jenis data untuk dibaca.

Wajib: `true`

Jenis: `string`

Nilai yang valid: `ascii, hex`

Pola yang valid: `^(ascii|hex|ASCII|HEX)$`

`id`

ID arbitrer untuk permintaan. Properti ini digunakan untuk memetakan permintaan input untuk respons output.

Wajib: `false`

Jenis: `string`

Pola yang valid: `.+`

Contoh input: Permintaan ASCII

```
{
    "data": "random serial data",
    "type": "ascii",
    "id": "abc123"
}
```

Contoh input: permintaan hex

```
{
    "data": "base64 encoded data",
    "type": "hex",
    "id": "abc123"
}
```

Data output

Konektor menerbitkan data output pada dua topik:

- Informasi status dari konektor pada `serial/+/status/#` topik.
- Respon dari permintaan baca pada `serial/+/read_response/#` topik.

Ketika menerbitkan topik ini, konektor menggantikan + wildcard dengan core nama sesuatu dan # wildcard dengan jalur ke port serial. Misalnya:

```
serial/core-thing-name/status/dev/serial-port
```

Filter topik: `serial/+/status/#`

Gunakan topik ini untuk mendengarkan status permintaan baca dan tulis. Jika sebuah `id` properti termasuk permintaan, itu dikembalikan dalam respon.

Contoh output: Sukses

```
{
    "response": {
        "status": "success"
    },
    "id": "abc123"
}
```

Contoh output: Gagal

Sebuah respon kegagalan mencakup sebuah `error_message` properti yang menjelaskan kesalahan atau timeout ditemui ketika melakukan operasi membaca atau menulis.

```
{
    "response": {
        "status": "fail",
        "error_message": "Could not write to port"
    },
    "id": "abc123"
}
```

Filter topik: `serial/+/read_response/#`

Gunakan topik ini untuk menerima data respon dari operasi baca. Data respon Base64 dikodekan jika jenis `hex`.

Contoh keluaran

```
{
    "data": "output of serial read operation"
    "id": "abc123"
```

```
}
```

Contoh Penggunaan

Gunakan langkah-langkah tingkat tinggi berikut untuk mengatur contoh fungsi Lambda Python 3.7 yang dapat Anda gunakan untuk mencoba konektor.

> **Note**
>
> - Jika Anda menggunakan waktu aktif Python lainnya, Anda dapat membuat symlink dari Python3.x ke Python 3.7.
> - Topik Memulai dengan konektor (konsol) dan Memulai dengan konektor (CLI) berisi langkah-langkah terperinci yang menunjukkan cara mengonfigurasi dan men-deploy contoh konektor Notifikasi Twilio.

1. Pastikan Anda memenuhi persyaratan untuk konektor.
2. Buat dan terbitkan fungsi Lambda yang mengirimkan data input ke konektor.

   Simpan kode contoh sebagai file PY. Unduh dan unzip AWS IoT Greengrass Core SDK for Python. Kemudian, buat paket zip yang berisi file PY dan folder `greengrasssdk` dalam tingkat root. Paket zip ini adalah paket deployment yang Anda unggah ke AWS Lambda.

   Setelah Anda membuat fungsi Lambda Python 3.7, terbitkankan versi fungsi dan buat alias.
3. Konfigurasikan grup Greengrass Anda.

   a. Tambahkan fungsi Lambda dengan aliasnya (direkomendasikan). Konfigurasikan siklus hidup Lambda sebagai berumur panjang (atau `"Pinned": true` dalam CLI).

   b. Tambahkan sumber daya perangkat lokal yang diperlukan dan berikan akses baca/tulis ke fungsi Lambda.

   c. Tambahkan konektor ke grup Anda dan konfigurasikan parameter.

   d. Tambahkan langganan ke grup yang memungkinkan konektor untuk menerima data input dan kirim data output pada filter topik yang didukung.

      - Atur fungsi Lambda sebagai sumber, konektor sebagai target, dan gunakan filter topik input yang mendukung.

- Atur konektor sebagai sumber, AWS IoT Core sebagai target, dan gunakan filter topik input yang mendukung. Anda menggunakan langganan ini untuk melihat pesan status dalam konsol AWS IoT tersebut.

4. Men-deploy grup.
5. Di konsol AWS IoT tersebut, pada halaman Tes ini, berlangganan ke topik data output untuk melihat pesan status dari konektor. Contoh fungsi Lambda yang berumur panjang dan mulai mengirim pesan segera setelah grup dalam-deploy.

   Setelah selesai pengujian, Anda dapat mengatur siklus hidup Lambda ke sesuai permintaan (atau `"Pinned": false` dalam CLI) dan men-deploy grup. Ini menghentikan fungsi dari mengirim pesan.

Contoh

Contoh fungsi Lambda berikut mengirimkan pesan input ke konektor.

```
import greengrasssdk
import json

TOPIC_REQUEST = 'serial/CORE_THING_NAME/write/dev/serial1'

# Creating a greengrass core sdk client
iot_client = greengrasssdk.client('iot-data')

def create_serial_stream_request():
 request = {
  "data": "TEST",
  "type": "ascii",
  "id": "abc123"
 }
 return request

def publish_basic_request():
 iot_client.publish(payload=json.dumps(create_serial_stream_request()),
 topic=TOPIC_REQUEST)

publish_basic_request()

def lambda_handler(event, context):
 return
```

Lisensi

Konektor Aliran Serial mencakup perangkat lunak/lisensi pihak ketiga berikut ini:

- pyserial/BSD

Konektor ini dirilis di bawah Perjanjian Lisensi Perangkat lunak core Greengrass.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi konektor.

Versi	Perubahan
3	Memperbarui waktu aktif Lambda untuk Python 3.7, yang mengubah persyaratan waktu aktif.
2	Tingkatkan konektor ARN untuk Wilayah AWS dukungan.
1	Pelepasan .

Sebuah grup Greengrass dapat berisi hanya satu versi konektor pada suatu waktu. Untuk informasi lebih lanjut tentang pembaruan versi konektor, lihat the section called “Versi upgrade konektor”.

Lihat juga

- Integrasikan dengan layanan dan protokol menggunakan konektor
- the section called “Memulai dengan konektor (konsol)”
- the section called “Memulai dengan konektor (CLI)”

ServiceNow MetricBase Konektor Integrasi

> **Warning**
>
> Konektor ini telah pindah kefase kehidupan yang panjang, danAWS IoT Greengrasstidak akan merilis pembaruan yang menyediakan fitur, penyempurnaan untuk fitur yang ada, patch

keamanan, atau perbaikan bug. Untuk informasi selengkapnya, lihat AWS IoT Greengrass Version 1kebijakan pemeliharaan.

Yang ServiceNow MetricBase Integrasipenghubungmenerbitkan metrik deret waktu dari perangkat Greengrass untuk ServiceNow MetricBase. Hal ini memungkinkan Anda untuk menyimpan, menganalisis, dan memvisualisasikan data time series dari lingkungan core Greengrass, dan bertindak pada peristiwa lokal.

Konektor ini menerima data deret waktu pada topik MQTT, dan menerbitkan data ke ServiceNow API secara berkala.

Anda dapat menggunakan konektor ini untuk mendukung skenario seperti:

- Buat peringatan dan alarm berbasis ambang batas berdasarkan data time series yang dikumpulkan dari perangkat Greengrass.
- Gunakan data layanan waktu dari perangkat Greengrass dengan aplikasi kustom yang dibangun pada ServiceNow platform.

Konektor ini memiliki versi berikut.

Versi	ARN
4	`arn:aws:greengrass:region::/connectors/ServiceNowMetricBaseIntegration/versions/4`
3	`arn:aws:greengrass:region::/connectors/ServiceNowMetricBaseIntegration/versions/3`
2	`arn:aws:greengrass:region::/connectors/ServiceNowMetricBaseIntegration/versions/2`
1	`arn:aws:greengrass:region::/connectors/ServiceNowMetricBaseIntegration/versions/1`

Untuk informasi tentang perubahan versi, lihat Changelog.

Persyaratan

Konektor ini memiliki persyaratan sebagai berikut:

Version 3 - 4

- AWS IoT Greengrass perangkat lunak Core v1.9.3 atau yang lebih baru. AWS IoT Greengrass harus dikonfigurasi untuk mendukung rahasia lokal, seperti yang dijelaskan dalam Persyaratan Rahasia.

  > **Note**
  >
  > Persyaratan ini mencakup mengizinkan akses ke rahasia Secrets Manager Anda. Jika Anda menggunakan peran layanan default Greengrass, Greengrass memiliki izin untuk mendapatkan nilai-nilai rahasia dengan nama yang dimulai dengan greengrass-.

- Python versi 3.7 atau 3.8 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.

  > **Note**
  >
  > Untuk menggunakan Python 3.8, jalankan perintah berikut untuk membuat link simbolik dari folder instalasi default Python 3.7 ke binari Python 3.8 yang diinstal.
  >
  > ```
  > sudo ln -s path-to-python-3.8/python3.8 /usr/bin/python3.7
  > ```
  >
  > Ini mengonfigurasi perangkat Anda untuk memenuhi persyaratan Python untuk AWS IoT Greengrass.

- SEBUAH ServiceNow akun dengan langganan yang diaktifkan untuk MetricBase. Sebagai tambahan, metrik dan tabel metrik harus dibuat dalam akun. Untuk informasi selengkapnya, lihatMetricBasedi dalam ServiceNow dokumentasi.
- Jenis teks rahasia diAWS Secrets Manageryang menyimpan nama pengguna dan kata sandi untuk masuk ke Anda ServiceNow misalnya dengan otentikasi dasar. Rahasianya harus berisi tombol "pengguna” dan "kata sandi” dengan nilai yang sesuai. Untuk informasi lebih lanjut, lihat Membuat rahasia dasar dalam AWS Secrets Manager Panduan Pengguna.

- Sebuah sumber daya rahasia dalam grup Greengrass yang mereferensikan rahasia Secrets Manager. Untuk informasi selengkapnya, lihat Men-deploy rahasia ke core.

Versions 1 - 2

- AWS IoT Greengrass perangkat lunak Core v1.7 atau yang lebih baru. AWS IoT Greengrass harus dikonfigurasi untuk mendukung rahasia lokal, seperti yang dijelaskan dalam Persyaratan Rahasia.

  > **Note**
  >
  > Persyaratan ini mencakup mengizinkan akses ke rahasia Secrets Manager Anda. Jika Anda menggunakan peran layanan default Greengrass, Greengrass memiliki izin untuk mendapatkan nilai-nilai rahasia dengan nama yang dimulai dengan greengrass-.

- Python versi 2.7 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.
- SEBUAH ServiceNow akun dengan langganan yang diaktifkan untuk MetricBase. Sebagai tambahan, metrik dan tabel metrik harus dibuat dalam akun. Untuk informasi selengkapnya, lihatMetricBasedi dalam ServiceNow dokumentasi.
- Jenis teks rahasia diAWS Secrets Manageryang menyimpan nama pengguna dan kata sandi untuk masuk ke Anda ServiceNow misalnya dengan otentikasi dasar. Rahasianya harus berisi tombol "pengguna" dan "kata sandi" dengan nilai yang sesuai. Untuk informasi lebih lanjut, lihat Membuat rahasia dasar dalam AWS Secrets Manager Panduan Pengguna.
- Sebuah sumber daya rahasia dalam grup Greengrass yang mereferensikan rahasia Secrets Manager. Untuk informasi selengkapnya, lihat Men-deploy rahasia ke core.

Parameter Konektor

Konektor ini menyediakan parameter berikut:

Version 4

`PublishInterval`

Jumlah maksimum detik untuk menunggu antara menerbitkan peristiwa untuk ServiceNow. Nilai maksimumnya adalah 900.

Konektor menerbitkan untuk ServiceNow ketika`PublishBatchSize`tercapai atau`PublishInterval`Kedaluwarsa.

Nama tampilan dalamAWS IoTKonsol: Publikasikan interval dalam hitungan detik

Wajib: `true`

Jenis: `string`

Nilai yang valid: `1 - 900`

Pola yang valid: `[1-9]|[1-9]\d|[1-9]\d\d|900`

`PublishBatchSize`

Jumlah maksimum nilai metrik yang dapat dikelompokkan sebelum dipublikasikan ServiceNow.

Konektor menerbitkan untuk ServiceNow ketika`PublishBatchSize`tercapai atau`PublishInterval`Kedaluwarsa.

Nama tampilan dalamAWS IoTKonsol: Ukuran batch publikasi batch publikasi

Wajib: `true`

Jenis: `string`

Pola yang valid: `^[0-9]+$`

`InstanceName`

Nama instans yang digunakan untuk terhubung ServiceNow.

Nama tampilan dalamAWS IoTKonsol: Nama dari ServiceNow contoh

Wajib: `true`

Jenis: `string`

Pola yang valid: `.+`

`DefaultTableName`

Nama tabel yang berisi`GlideRecord`terkait dengan deret waktu MetricBase basis data data data data Properti `table` dalam muatan pesan input dapat digunakan untuk menimpa nilai ini.

Nama tampilan dalamAWS IoTKonsol: Nama tabel untuk memuat metrik

Wajib: `true`

Jenis: `string`

Pola yang valid: `.+`

`MaxMetricsToRetain`

Jumlah maksimum metrik untuk dipertahankan dalam memori sebelum diganti dengan metrik baru.

Batas ini berlaku bila tidak ada koneksi ke internet dan konektor mulai buffer metrik untuk diterbitkan nanti. Ketika buffer penuh, metrik terlama digantikan oleh metrik baru.

> **Note**
>
> Metrik tidak dipertahankan jika proses host untuk konektor terganggu. Sebagai contoh, hal ini dapat terjadi selama deployment grup atau ketika perangkat restart.

Nilai ini harus lebih besar daripada ukuran batch dan cukup besar untuk menyimpan pesan berdasarkan tingkat masuk pesan MQTT.

Nama tampilan dalamAWS IoTKonsol: Metrik maksimum yang harus dipertahankan dalam memori

Wajib: `true`

Jenis: `string`

Pola yang valid: `^[0-9]+$`

`AuthSecretArn`

Rahasia dalamAWS Secrets Manageryang menyimpan ServiceNow nama pengguna dan kata sandi. Ini harus berupa rahasia jenis teks. Rahasianya harus berisi tombol "pengguna" dan "kata sandi" dengan nilai yang sesuai.

Nama tampilan dalamAWS IoTKonsol: ARN rahasia auth

Wajib: `true`

Jenis: `string`

Pola yang valid: `arn:aws:secretsmanager:[a-z0-9\-]+:[0-9]{12}:secret:([a-zA-Z0-9\\]+/)*[a-zA-Z0-9/_+=,.@\-]+-[a-zA-Z0-9]+`

`AuthSecretArn-ResourceId`

Sumber daya rahasia dalam grup yang mereferensikan rahasia Secrets Manager untuk ServiceNow kredenensi.

Nama tampilan dalamAWS IoTKonsol: Resource token Autentikasi token

Wajib: `true`

Jenis: `string`

Pola yang valid: `.+`

`IsolationMode`

Mode kontainerisasi untuk konektor ini. Default-nya adalah `GreengrassContainer`, yang berarti bahwa konektor berjalan dalam lingkungan waktu aktif terisolasi dalam kontainer AWS IoT Greengrass ini.

> **Note**
>
> Pengaturan kontainerisasi default untuk grup tidak berlaku untuk konektor.

Nama tampilan dalamAWS IoTKonsol: Mode isolasi kontainer

Wajib: `false`

Jenis: `string`

Nilai yang valid: `GreengrassContainer` or `NoContainer`

Pola yang valid: `^NoContainer$|^GreengrassContainer$`

Version 1 - 3

`PublishInterval`

Jumlah maksimum detik untuk menunggu antara menerbitkan peristiwa untuk ServiceNow. Nilai maksimumnya adalah 900.

Konektor menerbitkan untuk ServiceNow ketika`PublishBatchSize`tercapai atau`PublishInterval`Kedaluwarsa.

Nama tampilan dalamAWS IoTKonsol: Publikasikan interval dalam hitungan detik

Wajib: `true`

Jenis: `string`

Nilai yang valid: `1 - 900`

Pola yang valid: `[1-9]|[1-9]\d|[1-9]\d\d|900`

`PublishBatchSize`

Jumlah maksimum nilai metrik yang dapat dikelompokkan sebelum dipublikasikan ServiceNow.

Konektor menerbitkan untuk ServiceNow ketika`PublishBatchSize`tercapai atau`PublishInterval`Kedaluwarsa.

Nama tampilan dalamAWS IoTKonsol: Ukuran batch publikasi batch publikasi

Wajib: `true`

Jenis: `string`

Pola yang valid: `^[0-9]+$`

`InstanceName`

Nama instans yang digunakan untuk terhubung ServiceNow.

Nama tampilan dalamAWS IoTKonsol: Nama dari ServiceNow contoh

Wajib: `true`

Jenis: `string`

Pola yang valid: `.+`

`DefaultTableName`

Nama tabel yang berisi`GlideRecord`terkait dengan deret waktu MetricBase basis data data data data Properti `table` dalam muatan pesan input dapat digunakan untuk menimpa nilai ini.

Nama tampilan dalamAWS IoTKonsol: Nama tabel untuk memuat metrik

Wajib: `true`

Jenis: `string`

Pola yang valid: `.+`

`MaxMetricsToRetain`

Jumlah maksimum metrik untuk dipertahankan dalam memori sebelum diganti dengan metrik baru.

Batas ini berlaku bila tidak ada koneksi ke internet dan konektor mulai buffer metrik untuk diterbitkan nanti. Ketika buffer penuh, metrik terlama digantikan oleh metrik baru.

> **Note**
>
> Metrik tidak dipertahankan jika proses host untuk konektor terganggu. Sebagai contoh, hal ini dapat terjadi selama deployment grup atau ketika perangkat restart.

Nilai ini harus lebih besar daripada ukuran batch dan cukup besar untuk menyimpan pesan berdasarkan tingkat masuk pesan MQTT.

Nama tampilan dalamAWS IoTKonsol: Metrik maksimum yang harus dipertahankan dalam memori

Wajib: `true`

Jenis: `string`

Pola yang valid: `^[0-9]+$`

`AuthSecretArn`

Rahasia dalamAWS Secrets Manageryang menyimpan ServiceNow nama pengguna dan kata sandi. Ini harus berupa rahasia jenis teks. Rahasianya harus berisi tombol "pengguna" dan "kata sandi" dengan nilai yang sesuai.

Nama tampilan dalamAWS IoTKonsol: ARN rahasia auth

Wajib: `true`

Jenis: `string`

Pola yang valid: `arn:aws:secretsmanager:[a-z0-9\-]+:[0-9]{12}:secret:([a-zA-Z0-9\\]+/)*[a-zA-Z0-9/_+=,.@\-]+-[a-zA-Z0-9]+`

`AuthSecretArn-ResourceId`

Sumber daya rahasia dalam grup yang mereferensikan rahasia Secrets Manager untuk ServiceNow kredenensi.

Nama tampilan dalamAWS IoTKonsol: Resource token Autentikasi token

Wajib: `true`

Jenis: `string`

Pola yang valid: `.+`

Buat Contoh Konektor (AWS CLI)

Perintah CLI berikut membuat`ConnectorDefinition`dengan versi awal yang berisi ServiceNow MetricBase Konektor integrasi.

```
aws greengrass create-connector-definition --name MyGreengrassConnectors --initial-
version '{
    "Connectors": [
        {
            "Id": "MyServiceNowMetricBaseIntegrationConnector",
            "ConnectorArn": "arn:aws:greengrass:region::/connectors/
ServiceNowMetricBaseIntegration/versions/4",
            "Parameters": {
                "PublishInterval" : "10",
                "PublishBatchSize" : "50",
                "InstanceName" : "myinstance",
                "DefaultTableName" : "u_greengrass_app",
                "MaxMetricsToRetain" : "20000",
                "AuthSecretArn" : "arn:aws:secretsmanager:region:account-
id:secret:greengrass-secret-hash",
                "AuthSecretArn-ResourceId" : "MySecretResource",
                "IsolationMode" : "GreengrassContainer"
            }
        }
    ]
```

```
}'
```

> **Note**
>
> Fungsi Lambda dalam konektor mempunyai siklus hidup yang berumur panjang ini.

Di konsol AWS IoT Greengrass tersebut, Anda dapat menambahkan konektor dari halaman grup Konektor ini. Untuk informasi lebih lanjut, lihat the section called “Memulai dengan konektor (konsol)”.

Data input

Konektor ini menerima metrik deret waktu pada topik MQTT dan menerbitkan metrik untuk ServiceNow. Pesan input harus dalam format JSON.

Filter topik dalam langganan

`servicenow/metricbase/metric`

Properti pesan

`request`

Informasi tentang tabel, catatan, dan metrik. Permintaan ini mewakili `seriesRef` objek dalam permintaan time series POST. Untuk informasi lebih lanjut, lihat API Deret Waktu Clotho - POST.

Wajib: `true`

Jenis: `object` yang mencakup properti berikut:

`subject`

Ini `sys_id` adalah catatan khusus dalam tabel.

Wajib: `true`

Jenis: `string`

`metric_name`

Nama bidang metrik.

Wajib: `true`

Jenis: `string`

`table`

Nama tabel untuk menyimpan catatan dalam. Tentukan nilai ini untuk menimpa `DefaultTableName` parameter.

Wajib: `false`

Jenis: `string`

`value`

Nilai data poin individu.

Wajib: `true`

Jenis: `float`

`timestamp`

Timestamp dari data poin individu. Nilai default adalah waktu ketika ini.

Wajib: `false`

Jenis: `string`

Contoh input

```
{
    "request": {
        "subject":"ef43c6d40a0a0b5700c77f9bf387afe3",
        "metric_name":"u_count",
        "table": "u_greengrass_app"
        "value": 1.0,
        "timestamp": "2018-10-14T10:30:00"
    }
}
```

Data output

Konektor ini menerbitkan informasi status sebagai data output pada topik MQTT.

Filter topik dalam langganan

```
servicenow/metricbase/metric/status
```

Contoh output: Sukses

```
{
    "response": {
        "metric_name": "Errors",
        "table_name": "GliderProd",
        "processed_on": "2018-10-14T10:35:00",
        "response_id": "khjKSkj132qwr23fcba",
        "status": "success",
        "values": [
            {
                "timestamp": "2016-10-14T10:30:00",
                "value": 1.0
            },
            {
                "timestamp": "2016-10-14T10:31:00",
                "value": 1.1
            }
        ]
    }
}
```

Contoh output: Gagal

```
{
    "response": {
        "error": "InvalidInputException",
        "error_message": "metric value is invalid",
        "status": "fail"
    }
}
```

> **Note**
>
> Jika konektor mendeteksi kesalahan yang dapat diulang (sebagai contoh, kesalahan koneksi), konektor mengulang lagi publikasinya dalam batch berikutnya.

Contoh Penggunaan

Gunakan langkah-langkah tingkat tinggi berikut untuk mengatur contoh fungsi Lambda Python 3.7 yang dapat Anda gunakan untuk mencoba konektor.

> **Note**
>
> - Jika Anda menggunakan waktu aktif Python lainnya, Anda dapat membuat symlink dari Python3.x ke Python 3.7.
> - Topik Memulai dengan konektor (konsol) dan Memulai dengan konektor (CLI) berisi langkah-langkah terperinci yang menunjukkan cara mengonfigurasi dan men-deploy contoh konektor Notifikasi Twilio.

1. Pastikan Anda memenuhi persyaratan untuk konektor.
2. Buat dan terbitkan fungsi Lambda yang mengirimkan data input ke konektor.

   Simpan kode contoh sebagai file PY. Unduh dan unzip AWS IoT Greengrass Core SDK for Python. Kemudian, buat paket zip yang berisi file PY dan folder `greengrasssdk` dalam tingkat root. Paket zip ini adalah paket deployment yang Anda unggah ke AWS Lambda.

   Setelah Anda membuat fungsi Lambda Python 3.7, terbitkankan versi fungsi dan buat alias.
3. Konfigurasikan grup Greengrass Anda.

   a. Tambahkan fungsi Lambda dengan aliasnya (direkomendasikan). Konfigurasikan siklus hidup Lambda sebagai berumur panjang (atau `"Pinned": true` dalam CLI).

   b. Tambahkan sumber daya rahasia yang diperlukan dan berikan akses baca ke fungsi Lambda.

   c. Tambahkan konektor dan konfigurasikan parameter.

   d. Tambahkan langganan yang mengizinkan konektor untuk menerima data input dan mengirim data output pada filter topik yang didukung.

      - Atur fungsi Lambda sebagai sumber, konektor sebagai target, dan gunakan filter topik input yang mendukung.
      - Atur konektor sebagai sumber, AWS IoT Core sebagai target, dan gunakan filter topik input yang mendukung. Anda menggunakan langganan ini untuk melihat pesan status dalam konsol AWS IoT tersebut.

4. Men-deploy grup.
5. Di konsol AWS IoT tersebut, pada halaman Tes ini, berlangganan ke topik data output untuk melihat pesan status dari konektor. Contoh fungsi Lambda yang berumur panjang dan mulai mengirim pesan segera setelah grup dalam-deploy.

   Setelah selesai pengujian, Anda dapat mengatur siklus hidup Lambda ke sesuai permintaan (atau `"Pinned": false` dalam CLI) dan men-deploy grup. Ini menghentikan fungsi dari mengirim pesan.

Contoh

Contoh fungsi Lambda berikut mengirimkan pesan input ke konektor.

```
import greengrasssdk
import json

iot_client = greengrasssdk.client('iot-data')
SEND_TOPIC = 'servicenow/metricbase/metric'

def create_request_with_all_fields():
    return {
        "request": {
             "subject": '2efdf6badbd523803acfae441b961961',
             "metric_name": 'u_count',
             "value": 1234,
             "timestamp": '2018-10-20T20:22:20',
             "table": 'u_greengrass_metricbase_test'
        }
    }

def publish_basic_message():
    messageToPublish = create_request_with_all_fields()
    print("Message To Publish: ", messageToPublish)
    iot_client.publish(topic=SEND_TOPIC,
        payload=json.dumps(messageToPublish))

publish_basic_message()

def lambda_handler(event, context):
    return
```

Lisensi

Yang ServiceNow MetricBase Konektor integrasi mencakup perangkat lunak/lisensi pihak ketiga berikut:

- pysnow/MIT

Konektor ini dirilis di bawah Perjanjian Lisensi Perangkat lunak core Greengrass.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi konektor.

Versi	Perubahan
4	Ditambahkan parameter `IsolationMode` untuk mengonfigurasi mode kontainerisasi untuk konektor.
3	Memperbarui waktu aktif Lambda untuk Python 3.7, yang mengubah persyaratan waktu aktif.
2	Perbaiki untuk mengurangi pencatatan berlebihan.
1	Pelepasan .

Sebuah grup Greengrass dapat berisi hanya satu versi konektor pada suatu waktu. Untuk informasi lebih lanjut tentang pembaruan versi konektor, lihat the section called “Versi upgrade konektor”.

Lihat juga

- Integrasikan dengan layanan dan protokol menggunakan konektor
- the section called “Memulai dengan konektor (konsol)”
- the section called “Memulai dengan konektor (CLI)”

Konektor SNS

Sebuah konektor SNS menerbitkan pesan ke topik Amazon SNS. Hal ini memungkinkan server web, alamat email, dan pelanggan pesan lainnya untuk menanggapi peristiwa dalam grup Greengrass.

Konektor ini menerima informasi pesan SNS pada topik MQTT, dan kemudian mengirim pesan ke topik SNS tertentu. Anda dapat menggunakan fungsi Lambda kustom untuk implementasi penyaringan atau format logika pada pesan sebelum mereka diterbitkan untuk konektor ini.

Konektor ini memiliki versi berikut.

Versi	ARN
4	`arn:aws:greengrass:region::/connectors/SNS/versions/4`
3	`arn:aws:greengrass:region::/connectors/SNS/versions/3`
2	`arn:aws:greengrass:region::/connectors/SNS/versions/2`
1	`arn:aws:greengrass:region::/connectors/SNS/versions/1`

Untuk informasi tentang perubahan versi, lihat Changelog.

Persyaratan

Konektor ini memiliki persyaratan sebagai berikut:

Version 3 - 4

- AWS IoT Greengrass perangkat lunak Core v1.9.3 atau yang lebih baru.
- Python versi 3.7 atau 3.8 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.

> **Note**
>
> Untuk menggunakan Python 3.8, jalankan perintah berikut untuk membuat link simbolik dari folder instalasi default Python 3.7 ke binari Python 3.8 yang diinstal.
>
> ```
> sudo ln -s path-to-python-3.8/python3.8 /usr/bin/python3.7
> ```
>
> Ini mengonfigurasi perangkat Anda untuk memenuhi persyaratan Python untuk AWS IoT Greengrass.

- Sebuah topik SNS yang dikonfigurasi. Untuk informasi lebih lanjut, lihat Membuat topik Amazon SNS dalam Panduan Developer Amazon Simple Notification Service.
- Pada Peran grup Greengrass dikonfigurasi untuk mengizinkan `sns:Publish` tindakan pada target Amazon SNSstopic, seperti yang ditunjukkan contoh kebijakan IAM berikut.

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"Stmt1528133056761",
            "Action":[
                "sns:Publish"
            ],
            "Effect":"Allow",
            "Resource":[
                "arn:aws:sns:region:account-id:topic-name"
            ]
        }
    ]
}
```

Konektor ini mengizinkan Anda untuk secara dinamis menimpa topik default dalam muatan pesan input. Jika implementasi Anda menggunakan fitur ini, kebijakan IAM harus mengizinkan `sns:Publish` izin pada semua topik target. Anda dapat memberikan akses terperinci atau bersyarat ke sumber daya (sebagai contoh, dengan menggunakan skema penamaan wildcard *).

Untuk persyaratan peran grup, Anda harus mengonfigurasi peran untuk memberikan izin yang diperlukan dan memastikan peran telah ditambahkan ke grup. Untuk informasi lebih lanjut, lihat

the section called “Mengelola peran grup (konsol)” atau the section called “Kelola peran grup (CLI)”.

Versions 1 - 2

- AWS IoT Greengrass perangkat lunak Core v1.7 atau yang lebih baru.
- Python versi 2.7 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.
- Sebuah topik SNS yang dikonfigurasi. Untuk informasi lebih lanjut, lihat Membuat topik Amazon SNS dalam Panduan Developer Amazon Simple Notification Service.
- Pada Peran grup Greengrass dikonfigurasi untuk mengizinkan `sns:Publish` tindakan pada target Amazon SNSstopic, seperti yang ditunjukkan contoh kebijakan IAM berikut.

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"Stmt1528133056761",
            "Action":[
                "sns:Publish"
            ],
            "Effect":"Allow",
            "Resource":[
                "arn:aws:sns:region:account-id:topic-name"
            ]
        }
    ]
 }
```

Konektor ini mengizinkan Anda untuk secara dinamis menimpa topik default dalam muatan pesan input. Jika implementasi Anda menggunakan fitur ini, kebijakan IAM harus mengizinkan `sns:Publish` izin pada semua topik target. Anda dapat memberikan akses terperinci atau bersyarat ke sumber daya (sebagai contoh, dengan menggunakan skema penamaan wildcard *).

Untuk persyaratan peran grup, Anda harus mengonfigurasi peran untuk memberikan izin yang diperlukan dan memastikan peran telah ditambahkan ke grup. Untuk informasi lebih lanjut, lihat the section called “Mengelola peran grup (konsol)” atau the section called “Kelola peran grup (CLI)”.

Parameter Konektor

Konektor ini menyediakan parameter berikut:

Version 4

`DefaultSNSArn`

ARN topik SNS default untuk menerbitkan pesan ke. Topik tujuan dapat ditimpa oleh properti `sns_topic_arn` dalam muatan pesan input.

> **Note**
>
> Peran grup harus memberikan izin `sns:Publish` untuk semua topik target. Untuk informasi selengkapnya, lihat the section called “Persyaratan”.

Nama tampilan pada konsol AWS IoT tersebut: SNS Default ARN topik

Wajib: `true`

Jenis: `string`

Pola yang valid: `arn:aws:sns:([a-z]{2}-[a-z]+-\d{1}):(\d{12}):([a-zA-Z0-9-_]+)$`

`IsolationMode`

Mode kontainerisasi untuk konektor ini. Default-nya adalah `GreengrassContainer`, yang berarti bahwa konektor berjalan dalam lingkungan waktu aktif terisolasi dalam kontainer AWS IoT Greengrass ini.

> **Note**
>
> Pengaturan kontainerisasi default untuk grup tidak berlaku untuk konektor.

Nama tampilan pada konsol AWS IoT tersebut: Mode isolasi kontainer

Wajib: `false`

Jenis: `string`

Nilai yang valid: `GreengrassContainer` or `NoContainer`

Pola yang valid: `^NoContainer$|^GreengrassContainer$`

Versions 1 - 3

`DefaultSNSArn`

ARN topik SNS default untuk menerbitkan pesan ke. Topik tujuan dapat ditimpa oleh properti `sns_topic_arn` dalam muatan pesan input.

> **Note**
>
> Peran grup harus memberikan izin `sns:Publish` untuk semua topik target. Untuk informasi selengkapnya, lihat the section called “Persyaratan”.

Nama tampilan pada konsol AWS IoT tersebut: SNS Default ARN topik

Wajib: `true`

Jenis: `string`

Pola yang valid: `arn:aws:sns:([a-z]{2}-[a-z]+-\d{1}):(\d{12}):([a-zA-Z0-9-_]+)$`

Buat Contoh Konektor (AWS CLI)

Perintah CLI berikut membuat `ConnectorDefinition` dengan versi awal yang berisi konektor SNS.

```
aws greengrass create-connector-definition --name MyGreengrassConnectors --initial-
version '{
    "Connectors": [
        {
            "Id": "MySNSConnector",
            "ConnectorArn": "arn:aws:greengrass:region::/connectors/SNS/versions/4",
            "Parameters": {
                "DefaultSNSArn": "arn:aws:sns:region:account-id:topic-name",
                "IsolationMode" : "GreengrassContainer"
```

```
            }
        }
    ]
}'
```

Di konsol AWS IoT Greengrass tersebut, Anda dapat menambahkan konektor dari halaman grup Konektor ini. Untuk informasi lebih lanjut, lihat the section called “Memulai dengan konektor (konsol)”.

Data input

Konektor ini menerima informasi pesan SNS pada topik MQTT, dan kemudian menerbitkan pesan ke topik target SNS. Pesan input harus dalam format JSON.

Filter topik dalam langganan

`sns/message`

Properti pesan

`request`

Informasi tentang pesan yang akan dikirim ke topik SNS.

Wajib: `true`

Jenis: `object` yang mencakup properti berikut:

`message`

Isi pesan sebagai string atau dalam format JSON. Sebagai contoh, lihat Contoh input.

Untuk mengirim JSON, `message_structure` properti harus diatur ke `json` dan pesan harus berupa objek JSON string-encoded yang berisi `default` kunci.

Wajib: `true`

Jenis: `string`

Pola yang valid: `.*`

`subject`

Subjek pesan.

Wajib: `false`

Jenis: teks ASCII, hingga 100 karakter. Ini harus dimulai dengan tanda huruf, angka, atau tanda baca. Ini tidak harus mencakup jeda baris atau karakter kontrol.

Pola yang valid: `.*`

`sns_topic_arn`

ARN topik SNS untuk menerbitkan pesan ke. Jika ditentukan, konektor menerbitkan topik ini menggantikan topik default.

> **Note**
>
> Peran grup harus memberikan izin `sns:Publish` untuk setiap topik target. Untuk informasi selengkapnya, lihat the section called “Persyaratan”.

Wajib: `false`

Jenis: `string`

Pola yang valid: `arn:aws:sns:([a-z]{2}-[a-z]+-\d{1}):(\d{12}):([a-zA-Z0-9-_]+)$`

`message_structure`

Struktur pesan.

Diperlukan:`false`. Ini harus ditentukan untuk mengirim pesan JSON.

Tipe: `string`

Nilai yang valid: `json`

`id`

ID arbitrer untuk permintaan. Properti ini digunakan untuk memetakan permintaan input untuk respons output. Ketika ditentukan, `id` properti dalam objek respon diatur ke nilai ini. Jika Anda tidak menggunakan fitur ini, Anda dapat menghilangkan properti ini atau menentukan string kosong.

Wajib: `false`

Jenis: `string`

Pola yang valid: `.*`

Batas

Ukuran pesan dibatasi oleh ukuran maksimum pesan SNS 256 KB.

Contoh masukan: Pesan string

Contoh ini mengirimkan pesan string. Ini menentukan properti `sns_topic_arn` opsional, yang menimpa topik tujuan default.

```
{
    "request": {
        "subject": "Message subject",
        "message": "Message data",
        "sns_topic_arn": "arn:aws:sns:region:account-id:topic2-name"
    },
    "id": "request123"
}
```

Contoh masukan: pesan JSON

Contoh ini mengirimkan pesan sebagai string dikodekan objek JSON yang mencakup `default` kunci.

```
{
    "request": {
        "subject": "Message subject",
        "message": "{ \"default\": \"Message data\" }",
        "message_structure": "json"
    },
    "id": "request123"
}
```

Data output

Konektor ini menerbitkan informasi status sebagai data output pada topik MQTT.

Filter topik dalam langganan

`sns/message/status`

Contoh keluaran: Sukses

```
{
```

```
    "response": {
        "sns_message_id": "f80a81bc-f44c-56f2-a0f0-d5af6a727c8a",
        "status": "success"
    },
    "id": "request123"
}
```

Contoh keluaran: Kegagalan

```
{
   "response" : {
        "error": "InvalidInputException",
        "error_message": "SNS Topic Arn is invalid",
        "status": "fail"
   },
   "id": "request123"
}
```

Contoh Penggunaan

Gunakan langkah-langkah tingkat tinggi berikut untuk mengatur contoh fungsi Lambda Python 3.7 yang dapat Anda gunakan untuk mencoba konektor.

> **Note**
>
> - Jika Anda menggunakan waktu aktif Python lainnya, Anda dapat membuat symlink dari Python3.x ke Python 3.7.
> - Topik Memulai dengan konektor (konsol) dan Memulai dengan konektor (CLI) berisi langkah-langkah terperinci yang menunjukkan cara mengonfigurasi dan men-deploy contoh konektor Notifikasi Twilio.

1. Pastikan Anda memenuhi persyaratan untuk konektor.

   Untuk persyaratan peran grup, Anda harus mengonfigurasi peran untuk memberikan izin yang diperlukan dan memastikan peran telah ditambahkan ke grup. Untuk informasi lebih lanjut, lihat the section called “Mengelola peran grup (konsol)” atau the section called “Kelola peran grup (CLI)”.
2. Buat dan terbitkan fungsi Lambda yang mengirimkan data input ke konektor.

Simpan kode contoh sebagai file PY. Unduh dan unzip AWS IoT Greengrass Core SDK for Python. Kemudian, buat paket zip yang berisi file PY dan folder `greengrasssdk` dalam tingkat root. Paket zip ini adalah paket deployment yang Anda unggah ke AWS Lambda.

Setelah Anda membuat fungsi Lambda Python 3.7, terbitkankan versi fungsi dan buat alias.

3. Konfigurasikan grup Greengrass Anda.

   a. Tambahkan fungsi Lambda dengan aliasnya (direkomendasikan). Konfigurasikan siklus hidup Lambda sebagai berumur panjang (atau `"Pinned": true` dalam CLI).

   b. Tambahkan konektor dan konfigurasikan parameter.

   c. Tambahkan langganan yang mengizinkan konektor untuk menerima data input dan mengirim data output pada filter topik yang didukung.

      - Atur fungsi Lambda sebagai sumber, konektor sebagai target, dan gunakan filter topik input yang mendukung.
      - Atur konektor sebagai sumber, AWS IoT Core sebagai target, dan gunakan filter topik input yang mendukung. Anda menggunakan langganan ini untuk melihat pesan status dalam konsol AWS IoT tersebut.

4. Men-deploy grup.

5. Di konsol AWS IoT tersebut, pada halaman Tes ini, berlangganan ke topik data output untuk melihat pesan status dari konektor. Contoh fungsi Lambda yang berumur panjang dan mulai mengirim pesan segera setelah grup dalam-deploy.

   Setelah selesai pengujian, Anda dapat mengatur siklus hidup Lambda ke sesuai permintaan (atau `"Pinned": false` dalam CLI) dan men-deploy grup. Ini menghentikan fungsi dari mengirim pesan.

Contoh

Contoh fungsi Lambda berikut mengirimkan pesan input ke konektor.

```
import greengrasssdk
import time
import json

iot_client = greengrasssdk.client('iot-data')
send_topic = 'sns/message'
```

```
def create_request_with_all_fields():
    return  {
        "request": {
            "message": "Message from SNS Connector Test"
        },
        "id" : "req_123"
    }

def publish_basic_message():
    messageToPublish = create_request_with_all_fields()
    print("Message To Publish: ", messageToPublish)
    iot_client.publish(topic=send_topic,
        payload=json.dumps(messageToPublish))

publish_basic_message()

def lambda_handler(event, context):
    return
```

Lisensi

Konektor SNS termasuk perangkat lunak/lisensi pihak ketiga berikut:

- AWS SDK for Python (Boto3)/Lisensi 2.0 Apache
- botocore/Apache License 2.0
- dateutil/PSF License
- docutils/BSD License, Lisensi Publik Umum (GPL) GNU, Lisensi Dasar Perangkat Lunak Python, Domain Publik
- jmespath/MIT License
- s3transfer/Apache License 2.0
- urllib3/Lisensi MIT

Konektor ini dirilis di bawah Perjanjian Lisensi Perangkat lunak core Greengrass.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi konektor.

Versi	Perubahan
4	Ditambahkan parameter `IsolationMode` untuk mengonfigurasi mode kontainerisasi untuk konektor.
3	Memperbarui waktu aktif Lambda untuk Python 3.7, yang mengubah persyaratan waktu aktif.
2	Perbaiki untuk mengurangi pencatatan berlebihan.
1	Pelepasan awal.

Sebuah grup Greengrass dapat berisi hanya satu versi konektor pada suatu waktu. Untuk informasi lebih lanjut tentang pembaruan versi konektor, lihat the section called “Versi upgrade konektor”.

Lihat juga

- Integrasikan dengan layanan dan protokol menggunakan konektor
- the section called “Memulai dengan konektor (konsol)”
- the section called “Memulai dengan konektor (CLI)”
- Terbitkan tindakan dalam dokumentasi Boto 3
- Apakah Amazon Simple Notification Service? dalam Panduan Developer Amazon Simple Notification Service

Konektor Integrasi Splunk

> **Warning**
>
> Konektor ini telah pindah kefase kehidupan diperpanjang, danAWS IoT Greengrasstidak akan merilis pembaruan yang menyediakan fitur, penyempurnaan untuk fitur yang ada, patch keamanan, atau perbaikan bug. Untuk informasi selengkapnya, lihat AWS IoT Greengrass Version 1kebijakan pemeliharaan.

Konektor Integrasi Splunk menerbitkan data dari perangkat Greengrass ke Splunk. Ini mengizinkan Anda untuk menggunakan Splunk untuk memantau dan menganalisis lingkungan core Greengrass, dan bertindak pada peristiwa lokal. Konektor terintegrasi dengan HTTP Event Collector (HEC). Untuk informasi lebih lanjut, lihat Pengantar Splunk HTTP Event Collector dalam dokumentasi Splunk.

Konektor ini menerima data logging dan data peristiwa pada topik MQTT dan menerbitkan data seperti Splunk API.

Anda dapat menggunakan konektor ini untuk mendukung skenario industri, seperti:

- Operator dapat menggunakan data berkala dari aktuator dan sensor (sebagai contoh, suhu, tekanan, dan pembacaan air) untuk memulai alarm jika nilai melebihi ambang batas tertentu.
- Developer menggunakan data yang dikumpulkan dari mesin industri untuk membangun model ML yang dapat memantau peralatan untuk potensi masalah.

Konektor ini memiliki versi berikut.

Versi	ARN
4	`arn:aws:greengrass:` *`region`*`::/connectors/SplunkIntegration/versions/4`
3	`arn:aws:greengrass:` *`region`*`::/connectors/SplunkIntegration/versions/3`
2	`arn:aws:greengrass:` *`region`*`::/connectors/SplunkIntegration/versions/2`
1	`arn:aws:greengrass:` *`region`*`::/connectors/SplunkIntegration/versions/1`

Untuk informasi tentang perubahan versi, lihat Changelog.

Persyaratan

Konektor ini memiliki persyaratan sebagai berikut:

Version 3 - 4

- AWS IoT Greengrass perangkat lunak Core v1.9.3 atau yang lebih baru. AWS IoT Greengrass harus dikonfigurasi untuk mendukung rahasia lokal, seperti yang dijelaskan dalam Persyaratan Rahasia.

  > **Note**
  >
  > Persyaratan ini mencakup mengizinkan akses ke rahasia Secrets Manager Anda. Jika Anda menggunakan peran layanan default Greengrass, Greengrass memiliki izin untuk mendapatkan nilai-nilai rahasia dengan nama yang dimulai dengan greengrass-.

- Python versi 3.7 atau 3.8 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.

  > **Note**
  >
  > Untuk menggunakan Python 3.8, jalankan perintah berikut untuk membuat link simbolik dari folder instalasi default Python 3.7 ke binari Python 3.8 yang diinstal.
  >
  > ```
  > sudo ln -s path-to-python-3.8/python3.8 /usr/bin/python3.7
  > ```
  >
  > Ini mengonfigurasi perangkat Anda untuk memenuhi persyaratan Python untuk AWS IoT Greengrass.

- Fungsi HTTP Event Collector harus diaktifkan dalam Splunk. Untuk informasi lebih lanjut, lihat Atur dan gunakan HTTP eEvent Collector dalam Splunk Web dalam dokumentasi Splunk.
- Sebuah rahasia jenis teks dalam AWS Secrets Manager yang menyimpan token Splunk HTTP Event Collector Anda. Untuk informasi lebih lanjut, lihat Tentang token kolektor peristiwa dalam dokumentasi Splunk dan Membuat rahasia dasar dalam AWS Secrets Manager Panduan Pengguna.

> **Note**
>
> Untuk membuat rahasia dalam konsol Secrets Manager, masukkan token Anda pada tab Plaintext ini. Jangan sertakan tanda kutip atau pemformatan lainnya. Dalam API, tentukan token sebagai nilai untuk properti `SecretString` ini.

- Sebuah sumber daya rahasia dalam grup Greengrass yang mereferensikan rahasia Secrets Manager. Untuk informasi selengkapnya, lihat Men-deploy rahasia ke core.

Versions 1 - 2

- AWS IoT Greengrass perangkat lunak Core v1.7 atau yang lebih baru. AWS IoT Greengrass harus dikonfigurasi untuk mendukung rahasia lokal, seperti yang dijelaskan dalam Persyaratan Rahasia.

> **Note**
>
> Persyaratan ini mencakup mengizinkan akses ke rahasia Secrets Manager Anda. Jika Anda menggunakan peran layanan default Greengrass, Greengrass memiliki izin untuk mendapatkan nilai-nilai rahasia dengan nama yang dimulai dengan greengrass-.

- Python versi 2.7 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.
- Fungsi HTTP Event Collector harus diaktifkan dalam Splunk. Untuk informasi lebih lanjut, lihat Atur dan gunakan HTTP eEvent Collector dalam Splunk Web dalam dokumentasi Splunk.
- Sebuah rahasia jenis teks dalam AWS Secrets Manager yang menyimpan token Splunk HTTP Event Collector Anda. Untuk informasi lebih lanjut, lihat Tentang token kolektor peristiwa dalam dokumentasi Splunk dan Membuat rahasia dasar dalam AWS Secrets Manager Panduan Pengguna.

> **Note**
>
> Untuk membuat rahasia dalam konsol Secrets Manager, masukkan token Anda pada tab Plaintext ini. Jangan sertakan tanda kutip atau pemformatan lainnya. Dalam API, tentukan token sebagai nilai untuk properti `SecretString` ini.

- Sebuah sumber daya rahasia dalam grup Greengrass yang mereferensikan rahasia Secrets Manager. Untuk informasi selengkapnya, lihat Men-deploy rahasia ke core.

Parameter Konektor

Konektor ini menyediakan parameter berikut:

Version 4

`SplunkEndpoint`

Titik akhir dari instans Splunk Anda. Nilai ini harus berisi protokol, hostname, dan port.

Nama tampilan diAWS IoTKonsol: Titik akhir Splunk

Wajib: `true`

Jenis: `string`

Pola yang valid: `^(http:\/\/|https:\/\/)?[a-z0-9]+([-.]{1}[a-z0-9]+)*.[a-z]{2,5}(:[0-9]{1,5})?(\/.*)?$`

`MemorySize`

Jumlah memori (dalam KB) untuk mengalokasikan ke konektor.

Nama tampilan diAWS IoTKonsol: Ukuran memori

Wajib: `true`

Jenis: `string`

Pola yang valid: `^[0-9]+$`

`SplunkQueueSize`

Jumlah maksimum item untuk dipertahankan dalam memori sebelum item dikirimkan atau dibuang. Ketika batas ini terpenuhi, item tertua dalam antrian diganti dengan item yang lebih baru. Batas ini biasanya berlaku ketika tidak ada koneksi ke internet.

Nama tampilan diAWS IoTKonsol: Item maksimum untuk dipertahankan

Wajib: `true`

Jenis: `string`

Pola yang valid: `^[0-9]+$`

`SplunkFlushIntervalSeconds`

Interval (dalam detik) untuk menerbitkan data yang diterima ke Splunk HEC. Nilai maksimumnya adalah 900. Untuk mengonfigurasi konektor untuk menerbitkan item seperti yang diterima (tanpa batching), tentukan 0.

Nama tampilan diAWS IoTKonsol: Interval publikasi Splunk

Wajib: `true`

Jenis: `string`

Pola yang valid: `[0-9]|[1-9]\d|[1-9]\d\d|900`

`SplunkTokenSecretArn`

Rahasia dalam AWS Secrets Manager yang menyimpan token Splunk. Ini harus berupa rahasia jenis teks.

Nama tampilan diAWS IoTKonsol: ARN rahasia token autentikasi Splunk

Wajib: `true`

Jenis: `string`

Pola yang valid: `arn:aws:secretsmanager:[a-z]{2}-[a-z]+-\d{1}:\d{12}?:secret:[a-zA-Z0-9-_]+-[a-zA-Z0-9-_]+`

`SplunkTokenSecretArn-ResourceId`

Sumber daya rahasia dalam grup Greengrass yang mereferensikan rahasia Splunk.

Nama tampilan diAWS IoTKonsol: Sumber daya token autentikasi Splunk

Wajib: `true`

Jenis: `string`

Pola yang valid: `.+`

`SplunkCustomCALocation`

Jalur file kustom otoritas sertifikasi (CA) untuk Splunk (sebagai contoh, `/etc/ssl/certs/splunk.crt`).

Nama tampilan diAWS IoTKonsol: Lokasi otoritas sertifikat kustom Splunk

Wajib: `false`

Jenis: `string`

Pola yang valid: `^$|/.*`

`IsolationMode`

Mode kontainerisasi untuk konektor ini. Default-nya adalah `GreengrassContainer`, yang berarti bahwa konektor berjalan dalam lingkungan waktu aktif terisolasi dalam kontainer AWS IoT Greengrass ini.

> **Note**
>
> Pengaturan kontainerisasi default untuk grup tidak berlaku untuk konektor.

Nama tampilan diAWS IoTKonsol: Mode isolasi kontainer

Wajib: `false`

Jenis: `string`

Nilai yang valid: `GreengrassContainer` or `NoContainer`

Pola yang valid: `^NoContainer$|^GreengrassContainer$`

Version 1 - 3

`SplunkEndpoint`

Titik akhir dari instans Splunk Anda. Nilai ini harus berisi protokol, hostname, dan port.

Nama tampilan diAWS IoTKonsol: Titik akhir Splunk

Wajib: `true`

Jenis: `string`

Pola yang valid: `^(http:\/\/|https:\/\/)?[a-z0-9]+([-.]{1}[a-z0-9]+)*.[a-z]{2,5}(:[0-9]{1,5})?(\/.*)?$`

`MemorySize`

Jumlah memori (dalam KB) untuk mengalokasikan ke konektor.

Nama tampilan diAWS IoTKonsol: Ukuran memori

Wajib: `true`

Jenis: `string`

Pola yang valid: `^[0-9]+$`

`SplunkQueueSize`

Jumlah maksimum item untuk dipertahankan dalam memori sebelum item dikirimkan atau dibuang. Ketika batas ini terpenuhi, item tertua dalam antrian diganti dengan item yang lebih baru. Batas ini biasanya berlaku ketika tidak ada koneksi ke internet.

Nama tampilan diAWS IoTKonsol: Item maksimum untuk dipertahankan

Wajib: `true`

Jenis: `string`

Pola yang valid: `^[0-9]+$`

`SplunkFlushIntervalSeconds`

Interval (dalam detik) untuk menerbitkan data yang diterima ke Splunk HEC. Nilai maksimumnya adalah 900. Untuk mengonfigurasi konektor untuk menerbitkan item seperti yang diterima (tanpa batching), tentukan 0.

Nama tampilan diAWS IoTKonsol: Interval publikasi Splunk

Wajib: `true`

Jenis: `string`

Pola yang valid: `[0-9]|[1-9]\d|[1-9]\d\d|900`

`SplunkTokenSecretArn`

Rahasia dalam AWS Secrets Manager yang menyimpan token Splunk. Ini harus berupa rahasia jenis teks.

Nama tampilan diAWS IoTKonsol: ARN rahasia token autentikasi Splunk

Wajib: `true`

Jenis: `string`

Pola yang valid: `arn:aws:secretsmanager:[a-z]{2}-[a-z]+-\d{1}:\d{12}?:secret:[a-zA-Z0-9-_]+-[a-zA-Z0-9-_]+`

`SplunkTokenSecretArn-ResourceId`

Sumber daya rahasia dalam grup Greengrass yang mereferensikan rahasia Splunk.

Nama tampilan diAWS IoTKonsol: Sumber daya token autentikasi Splunk

Wajib: `true`

Jenis: `string`

Pola yang valid: `.+`

`SplunkCustomCALocation`

Jalur file kustom otoritas sertifikasi (CA) untuk Splunk (sebagai contoh, `/etc/ssl/certs/splunk.crt`).

Nama tampilan diAWS IoTKonsol: Lokasi otoritas sertifikat kustom Splunk

Wajib: `false`

Jenis: `string`

Pola yang valid: `^$|/.*`

Buat Contoh Konektor (AWS CLI)

Perintah CLI berikut membuat `ConnectorDefinition` dengan versi awal yang berisi konektor Integrasi Splunk.

```
aws greengrass create-connector-definition --name MyGreengrassConnectors --initial-
version '{
    "Connectors": [
        {
            "Id": "MySplunkIntegrationConnector",
```

```
            "ConnectorArn": "arn:aws:greengrass:region::/connectors/SplunkIntegration/
versions/4",
            "Parameters": {
                "SplunkEndpoint": "https://myinstance.cloud.splunk.com:8088",
                "MemorySize": 200000,
                "SplunkQueueSize": 10000,
                "SplunkFlushIntervalSeconds": 5,
                "SplunkTokenSecretArn":"arn:aws:secretsmanager:region:account-
id:secret:greengrass-secret-hash",
                "SplunkTokenSecretArn-ResourceId": "MySplunkResource",
                "IsolationMode" : "GreengrassContainer"
            }
        }
    ]
}'
```

> **Note**
>
> Fungsi Lambda dalam konektor mempunyai siklus hidup yang berumur panjang ini.

Di konsol AWS IoT Greengrass tersebut, Anda dapat menambahkan konektor dari halaman grup Konektor ini. Untuk informasi lebih lanjut, lihat the section called “Memulai dengan konektor (konsol)”.

Data input

Konektor ini menerima data logging dan peristiwa pada topik MQTT dan menerbitkan data yang diterima seperti Splunk API. Pesan input harus dalam format JSON.

Filter topik dalam langganan

`splunk/logs/put`

Properti pesan

`request`

Data peristiwa untuk mengirim ke Splunk API. Acara harus memenuhi spesifikasi dari layanan/kolektor API.

Wajib: `true`

Jenis: `object`. Hanya`event`properti diperlukan.

`id`

ID arbitrer untuk permintaan. Properti ini digunakan untuk memetakan permintaan input untuk status output.

Wajib: `false`

Jenis: `string`

Batasan

Semua batas yang dikenakan oleh Splunk API berlaku ketika menggunakan konektor ini. Untuk informasi lebih lanjut, lihat layanan/kolektor.

Contoh input

```
{
    "request": {
        "event": "some event",
        "fields": {
            "severity": "INFO",
            "category": [
                "value1",
                "value2"
            ]
        }
    },
    "id": "request123"
}
```

Data output

Konektor ini menerbitkan data output pada dua topik:

- Informasi status pada `splunk/logs/put/status` topik.
- Kesalahan pada topik `splunk/logs/put/error` ini.

Filter topik: `splunk/logs/put/status`

Gunakan topik ini untuk mendengarkan status permintaan. Setiap kali konektor mengirimkan batch data yang diterima ke Splunk API, Splunk API menerbitkan daftar ID permintaan yang berhasil dan gagal.

Contoh keluaran

```
{
    "response": {
        "succeeded": [
            "request123",
            ...
        ],
        "failed": [
            "request789",
            ...
        ]
    }
}
```

Filter topik: `splunk/logs/put/error`

Gunakan topik ini untuk mendengarkan kesalahan dari konektor. Properti `error_message` yang menjelaskan kesalahan atau timeout yang dihadapi sambil memproses permintaan.

Contoh keluaran

```
{
    "response": {
        "error": "UnauthorizedException",
        "error_message": "invalid splunk token",
        "status": "fail"
    }
}
```

> **Note**
>
> Jika konektor mendeteksi kesalahan yang dapat diulang (sebagai contoh, kesalahan koneksi), konektor mengulang lagi publikasinya dalam batch berikutnya.

Contoh Penggunaan

Gunakan langkah-langkah tingkat tinggi berikut untuk mengatur contoh fungsi Lambda Python 3.7 yang dapat Anda gunakan untuk mencoba konektor.

> **Note**
>
> - Jika Anda menggunakan waktu aktif Python lainnya, Anda dapat membuat symlink dari Python3.x ke Python 3.7.
> - Topik Memulai dengan konektor (konsol) dan Memulai dengan konektor (CLI) berisi langkah-langkah terperinci yang menunjukkan cara mengonfigurasi dan men-deploy contoh konektor Notifikasi Twilio.

1. Pastikan Anda memenuhi persyaratan untuk konektor.
2. Buat dan terbitkan fungsi Lambda yang mengirimkan data input ke konektor.

   Simpan kode contoh sebagai file PY. Unduh dan unzip AWS IoT Greengrass Core SDK for Python. Kemudian, buat paket zip yang berisi file PY dan folder `greengrasssdk` dalam tingkat root. Paket zip ini adalah paket deployment yang Anda unggah ke AWS Lambda.

   Setelah Anda membuat fungsi Lambda Python 3.7, terbitkankan versi fungsi dan buat alias.
3. Konfigurasikan grup Greengrass Anda.

   a. Tambahkan fungsi Lambda dengan aliasnya (direkomendasikan). Konfigurasikan siklus hidup Lambda sebagai berumur panjang (atau `"Pinned": true` dalam CLI).

   b. Tambahkan sumber daya rahasia yang diperlukan dan berikan akses baca ke fungsi Lambda.

   c. Tambahkan konektor dan konfigurasikan parameter.

   d. Tambahkan langganan yang mengizinkan konektor untuk menerima data input dan mengirim data output pada filter topik yang didukung.

      - Atur fungsi Lambda sebagai sumber, konektor sebagai target, dan gunakan filter topik input yang mendukung.
      - Atur konektor sebagai sumber, AWS IoT Core sebagai target, dan gunakan filter topik input yang mendukung. Anda menggunakan langganan ini untuk melihat pesan status dalam konsol AWS IoT tersebut.
4. Men-deploy grup.
5. Di konsol AWS IoT tersebut, pada halaman Tes ini, berlangganan ke topik data output untuk melihat pesan status dari konektor. Contoh fungsi Lambda yang berumur panjang dan mulai mengirim pesan segera setelah grup dalam-deploy.

Setelah selesai pengujian, Anda dapat mengatur siklus hidup Lambda ke sesuai permintaan (atau `"Pinned": false` dalam CLI) dan men-deploy grup. Ini menghentikan fungsi dari mengirim pesan.

Contoh

Contoh fungsi Lambda berikut mengirimkan pesan input ke konektor.

```
import greengrasssdk
import time
import json

iot_client = greengrasssdk.client('iot-data')
send_topic = 'splunk/logs/put'

def create_request_with_all_fields():
    return {
        "request": {
            "event": "Access log test message."
        },
        "id" : "req_123"
    }

def publish_basic_message():
    messageToPublish = create_request_with_all_fields()
    print("Message To Publish: ", messageToPublish)
    iot_client.publish(topic=send_topic,
        payload=json.dumps(messageToPublish))

publish_basic_message()

def lambda_handler(event, context):
    return
```

Lisensi

Konektor ini dirilis di bawah Perjanjian Lisensi Perangkat lunak core Greengrass.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi konektor.

Versi	Perubahan
4	Ditambahkan parameter `IsolationMode` untuk mengonfigurasi mode kontainerisasi untuk konektor.
3	Memperbarui waktu aktif Lambda untuk Python 3.7, yang mengubah persyaratan waktu aktif.
2	Perbaiki untuk mengurangi pencatatan berlebihan.
1	Pelepasan .

Sebuah grup Greengrass dapat berisi hanya satu versi konektor pada suatu waktu. Untuk informasi lebih lanjut tentang pembaruan versi konektor, lihat the section called “Versi upgrade konektor”.

Lihat juga

- Integrasikan dengan layanan dan protokol menggunakan konektor
- the section called “Memulai dengan konektor (konsol)”
- the section called “Memulai dengan konektor (CLI)”

Konektor Notifikasi Twilio

> **Warning**
>
> Konektor ini telah pindah kefase kehidupan yang panjang, danAWS IoT Greengrasstidak akan merilis pembaruan yang menyediakan fitur, penyempurnaan untuk fitur yang ada, patch keamanan, atau perbaikan bug. Untuk informasi selengkapnya, lihat AWS IoT Greengrass Version 1kebijakan pemeliharaan.

Konektor Notifikasi Twilio melakukan panggilan telepon otomatis atau mengirim pesan teks melalui Twilio. Anda dapat menggunakan konektor ini untuk mengirim notifikasi dalam menanggapi peristiwa dalam grup Greengrass. Untuk panggilan telepon, konektor dapat meneruskan pesan suara ke penerima.

Konektor ini menerima informasi pesan Twilio pada topik MQTT, dan kemudian memicu notifikasi Twilio.

> **Note**
>
> Untuk tutorial yang menunjukkan cara menggunakan konektor notifikasi Twilio, lihat the section called “Memulai dengan konektor (konsol)” atau the section called “Memulai dengan konektor (CLI)”.

Konektor ini memiliki versi berikut.

Versi	ARN
5	`arn:aws:greengrass:region::/connectors/TwilioNotifications/versions/5`
4	`arn:aws:greengrass:region::/connectors/TwilioNotifications/versions/4`
3	`arn:aws:greengrass:region::/connectors/TwilioNotifications/versions/3`
2	`arn:aws:greengrass:region::/connectors/TwilioNotifications/versions/2`
1	`arn:aws:greengrass:region::/connectors/TwilioNotifications/versions/1`

Untuk informasi tentang perubahan versi, lihat Changelog.

Persyaratan

Konektor ini memiliki persyaratan sebagai berikut:

Version 4 - 5

- AWS IoT Greengrass perangkat lunak Core v1.9.3 atau yang lebih baru. AWS IoT Greengrass harus dikonfigurasi untuk mendukung rahasia lokal, seperti yang dijelaskan dalam Persyaratan Rahasia.

  > **Note**
  >
  > Persyaratan ini mencakup mengizinkan akses ke rahasia Secrets Manager Anda. Jika Anda menggunakan peran layanan default Greengrass, Greengrass memiliki izin untuk mendapatkan nilai-nilai rahasia dengan nama yang dimulai dengan greengrass-.

- Python versi 3.7 atau 3.8 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.

  > **Note**
  >
  > Untuk menggunakan Python 3.8, jalankan perintah berikut untuk membuat link simbolik dari folder instalasi default Python 3.7 ke binari Python 3.8 yang diinstal.
  >
  > ```
  > sudo ln -s path-to-python-3.8/python3.8 /usr/bin/python3.7
  > ```
  >
  > Ini mengonfigurasi perangkat Anda untuk memenuhi persyaratan Python untuk AWS IoT Greengrass.

- Twilio account SID, token auth, dan nomor telepon berkemampuan Twilio. Setelah Anda membuat proyek Twilio, nilai-nilai ini tersedia dalam dasbor proyek.

  > **Note**
  >
  > Anda dapat menggunakan akun percobaan Twilio. Jika menggunakan akun uji coba, Anda harus menambahkan nomor telepon penerima selain Twilio ke daftar nomor telepon terverifikasi. Untuk informasi lebih lanjut, lihat Cara Bekerja dengan Akun Trial Twilio Gratis.

- Sebuah rahasia jenis teks dalam AWS Secrets Manager yang menyimpan token auth Twilio. Untuk informasi lebih lanjut, lihat Membuat rahasia dasar dalam AWS Secrets Manager Panduan Pengguna.

  > **Note**
  >
  > Untuk membuat rahasia dalam konsol Secrets Manager, masukkan token Anda pada tab Plaintext ini. Jangan sertakan tanda kutip atau pemformatan lainnya. Dalam API, tentukan token sebagai nilai untuk properti `SecretString` ini.

- Sebuah sumber daya rahasia dalam grup Greengrass yang mereferensikan rahasia Secrets Manager. Untuk informasi selengkapnya, lihat Men-deploy rahasia ke core.

Versions 1 - 3

- AWS IoT Greengrass perangkat lunak Core v1.7 atau yang lebih baru. AWS IoT Greengrass harus dikonfigurasi untuk mendukung rahasia lokal, seperti yang dijelaskan dalam Persyaratan Rahasia.

  > **Note**
  >
  > Persyaratan ini mencakup mengizinkan akses ke rahasia Secrets Manager Anda. Jika Anda menggunakan peran layanan default Greengrass, Greengrass memiliki izin untuk mendapatkan nilai-nilai rahasia dengan nama yang dimulai dengan greengrass-.

- Python versi 2.7 diinstal pada perangkat core dan ditambahkan ke variabel lingkungan PATH.
- Twilio account SID, token auth, dan nomor telepon berkemampuan Twilio. Setelah Anda membuat proyek Twilio, nilai-nilai ini tersedia dalam dasbor proyek.

  > **Note**
  >
  > Anda dapat menggunakan akun percobaan Twilio. Jika menggunakan akun uji coba, Anda harus menambahkan nomor telepon penerima selain Twilio ke daftar nomor telepon terverifikasi. Untuk informasi lebih lanjut, lihat Cara Bekerja dengan Akun Trial Twilio Gratis.

- Sebuah rahasia jenis teks dalam AWS Secrets Manager yang menyimpan token auth Twilio. Untuk informasi lebih lanjut, lihat Membuat rahasia dasar dalam AWS Secrets Manager Panduan Pengguna.

  > **Note**
  >
  > Untuk membuat rahasia dalam konsol Secrets Manager, masukkan token Anda pada tab Plaintext ini. Jangan sertakan tanda kutip atau pemformatan lainnya. Dalam API, tentukan token sebagai nilai untuk properti `SecretString` ini.

- Sebuah sumber daya rahasia dalam grup Greengrass yang mereferensikan rahasia Secrets Manager. Untuk informasi selengkapnya, lihat Men-deploy rahasia ke core.

Parameter Konektor

Konektor ini menyediakan parameter berikut.

Version 5

`TWILIO_ACCOUNT_SID`

Akun Twilio SID yang digunakan untuk memanggil Twilio API.

Nama tampilanAWS IoTKonsol: Akun Twilio

Wajib: `true`

Jenis: `string`

Pola yang valid: `.+`

`TwilioAuthTokenSecretArn`

ARN rahasia Secrets Manager yang menyimpan token auth Twilio.

> **Note**
>
> Ini digunakan untuk mengakses nilai rahasia lokal pada core.

Nama tampilanAWS IoTKonsol: ARN dari Twilio auth token rahasia

Wajib: `true`

Jenis: `string`

Pola yang valid: `arn:aws:secretsmanager:[a-z0-9\-]+:[0-9]{12}:secret:([a-zA-Z0-9\\]+/)*[a-zA-Z0-9/_+=,.@\-]+-[a-zA-Z0-9]+`

`TwilioAuthTokenSecretArn-ResourceId`

ID dari sumber rahasia dalam grup Greengrass yang mereferensikan rahasia untuk token auth Twilio.

Nama tampilanAWS IoTKonsol: Sumber daya token autentikasi Twilio

Wajib: `true`

Jenis: `string`

Pola yang valid: `.+`

`DefaultFromPhoneNumber`

Nomor telepon Twilio diaktifkan default yang Twilio gunakan untuk mengirim pesan. Twilio menggunakan nomor ini untuk memulai teks atau panggilan.

- Jika Anda tidak mengonfigurasi nomor telepon default, Anda harus menentukan nomor telepon dalam `from_number` properti dalam tubuh pesan input.
- Jika Anda mengonfigurasi nomor telepon default, Anda dapat mengganti default secara opsional dengan menentukan `from_number` properti dalam tubuh pesan input.

Nama tampilanAWS IoTKonsol: Nomor telepon

Wajib: `false`

Jenis: `string`

Pola yang valid: `^$|\+[0-9]+`

`IsolationMode`

Mode kontainerisasi untuk konektor ini. Default-nya adalah `GreengrassContainer`, yang berarti bahwa konektor berjalan dalam lingkungan waktu aktif terisolasi dalam kontainer AWS IoT Greengrass ini.

> **Note**
>
> Pengaturan kontainerisasi default untuk grup tidak berlaku untuk konektor.

Nama tampilanAWS IoTKonsol: Mode isolasi kontainer

Wajib: `false`

Jenis: `string`

Nilai yang valid: `GreengrassContainer` or `NoContainer`

Pola yang valid: `^NoContainer$|^GreengrassContainer$`

Version 1 - 4

`TWILIO_ACCOUNT_SID`

Akun Twilio SID yang digunakan untuk memanggil Twilio API.

Nama tampilanAWS IoTKonsol: Akun Twilio

Wajib: `true`

Jenis: `string`

Pola yang valid: `.+`

`TwilioAuthTokenSecretArn`

ARN rahasia Secrets Manager yang menyimpan token auth Twilio.

> **Note**
>
> Ini digunakan untuk mengakses nilai rahasia lokal pada core.

Nama tampilanAWS IoTKonsol: ARN dari Twilio auth token rahasia

Wajib: `true`

Jenis: `string`

Pola yang valid: `arn:aws:secretsmanager:[a-z0-9\-]+:[0-9]{12}:secret:([a-zA-Z0-9\\]+/)*[a-zA-Z0-9/_+=,.@\-]+-[a-zA-Z0-9]+`

`TwilioAuthTokenSecretArn-ResourceId`

ID dari sumber rahasia dalam grup Greengrass yang mereferensikan rahasia untuk token auth Twilio.

Nama tampilanAWS IoTKonsol: Sumber daya token autentikasi Twilio

Wajib: `true`

Jenis: `string`

Pola yang valid: `.+`

`DefaultFromPhoneNumber`

Nomor telepon Twilio diaktifkan default yang Twilio gunakan untuk mengirim pesan. Twilio menggunakan nomor ini untuk memulai teks atau panggilan.

- Jika Anda tidak mengonfigurasi nomor telepon default, Anda harus menentukan nomor telepon dalam `from_number` properti dalam tubuh pesan input.
- Jika Anda mengonfigurasi nomor telepon default, Anda dapat mengganti default secara opsional dengan menentukan `from_number` properti dalam tubuh pesan input.

Nama tampilanAWS IoTKonsol: Nomor telepon

Wajib: `false`

Jenis: `string`

Pola yang valid: `^$|\+[0-9]+`

Buat Contoh Konektor (AWS CLI)

Contoh berikut perintah CLI membuat `ConnectorDefinition` dengan versi awal yang berisi konektor Notifikasi Twilio.

```
aws greengrass create-connector-definition --name MyGreengrassConnectors --initial-
version '{
    "Connectors": [
        {
```

```
            "Id": "MyTwilioNotificationsConnector",
            "ConnectorArn": "arn:aws:greengrass:region::/connectors/
TwilioNotifications/versions/5",
            "Parameters": {
                "TWILIO_ACCOUNT_SID": "abcd12345xyz",
                "TwilioAuthTokenSecretArn": "arn:aws:secretsmanager:region:account-
id:secret:greengrass-secret-hash",
                "TwilioAuthTokenSecretArn-ResourceId": "MyTwilioSecret",
                "DefaultFromPhoneNumber": "+19999999999",
                "IsolationMode" : "GreengrassContainer"
            }
        }
    ]
}'
```

Untuk tutorial yang menunjukkan cara menambahkan konektor Notifikasi Twilio ke grup, lihat the section called “Memulai dengan konektor (CLI)” dan the section called “Memulai dengan konektor (konsol)”.

Data input

Konektor ini menerima informasi pesan Twilio pada dua topik MQTT. Pesan input harus dalam format JSON.

- Informasi pesan teks pada `twilio/txt` topik.
- Informasi pesan telepon dalam `twilio/call` topik.

> **Note**
>
> Muatan pesan input dapat mencakup pesan teks (`message`) atau pesan suara (`voice_message_location`), tapi tidak keduanya.

Filter topik:**`twilio/txt`**

Properti pesan

`request`

Informasi tentang Notifikasi Twilio.

Wajib: `true`

Jenis: `object` yang mencakup properti berikut:

`recipient`

Penerima pesan. Hanya satu penerima yang didukung.

Wajib: `true`

Jenis: `object` yang mencakup properti berikut:

`name`

Nama penerima.

Wajib: `true`

Jenis: `string`

Pola yang valid: `.*`

`phone_number`

Nomor telepon penerima.

Wajib: `true`

Jenis: `string`

Pola yang valid: `\+[1-9]+`

`message`

Konten teks dari pesan teks. Hanya pesan teks yang didukung pada topik ini. Untuk pesan suara, gunakan `twilio/call`.

Wajib: `true`

Jenis: `string`

Pola yang valid: `.+`

`from_number`

Nomor telepon pengirim. Twilio menggunakan nomor telepon ini untuk memulai pesan. Properti ini diperlukan jika parameter `DefaultFromPhoneNumber` tidak dikonfigurasi. Jika `DefaultFromPhoneNumber` dikonfigurasi, Anda dapat menggunakan properti ini untuk menimpa default.

Wajib: `false`

Jenis: `string`

Pola yang valid: `\+[1-9]+`

`retries`

Nomor retries. Default-nya adalah 0.

Wajib: `false`

Jenis: `integer`

`id`

ID arbitrer untuk permintaan. Properti ini digunakan untuk memetakan permintaan input untuk respons output.

Wajib: `true`

Jenis: `string`

Pola yang valid: `.+`

Contoh input

```
{
    "request": {
        "recipient": {
            "name": "Darla",
            "phone_number": "+12345000000",
            "message": "Hello from the edge"
        },
        "from_number": "+19999999999",
        "retries": 3
    },
    "id": "request123"
}
```

Filter topik:**`twilio/call`**

Properti pesan

`request`

Informasi tentang Notifikasi Twilio.

Wajib: `true`

Jenis: `object` yang mencakup properti berikut:

`recipient`

Penerima pesan. Hanya satu penerima yang didukung.

Wajib: `true`

Jenis: `object` yang mencakup properti berikut:

`name`

Nama penerima.

Wajib: `true`

Jenis: `string`

Pola yang valid: `.+`

`phone_number`

Nomor telepon penerima.

Wajib: `true`

Jenis: `string`

Pola yang valid: `\+[1-9]+`

`voice_message_location`

URL konten audio untuk pesan suara. Ini harus dalam format TWIML. Hanya pesan suara yang didukung pada topik ini. Untuk pesan teks, gunakan `twilio/txt`.

Wajib: `true`

Jenis: `string`

Pola yang valid: `.+`

`from_number`

Nomor telepon pengirim. Twilio menggunakan nomor telepon ini untuk memulai pesan. Properti ini diperlukan jika parameter `DefaultFromPhoneNumber` tidak dikonfigurasi.

Jika `DefaultFromPhoneNumber` dikonfigurasi, Anda dapat menggunakan properti ini untuk menimpa default.

Wajib: `false`

Jenis: `string`

Pola yang valid: `\+[1-9]+`

`retries`

Nomor retries. Default-nya adalah 0.

Wajib: `false`

Jenis: `integer`

`id`

ID arbitrer untuk permintaan. Properti ini digunakan untuk memetakan permintaan input untuk respons output.

Wajib: `true`

Jenis: `string`

Pola yang valid: `.+`

Contoh input

```
{
    "request": {
        "recipient": {
            "name": "Darla",
            "phone_number": "+12345000000",
            "voice_message_location": "https://some-public-TwiML"
        },
        "from_number": "+19999999999",
        "retries": 3
    },
    "id": "request123"
}
```

Data output

Konektor ini menerbitkan informasi status sebagai data output pada topik MQTT.

Filter topik dalam langganan

```
twilio/message/status
```

Contoh output: Sukses

```
{
    "response": {
        "status": "success",
        "payload": {
            "from_number": "+19999999999",
            "messages": {
                "message_status": "queued",
                "to_number": "+12345000000",
                "name": "Darla"
            }
        }
    },
    "id": "request123"
}
```

Contoh output: Gagal

```
{
    "response": {
        "status": "fail",
        "error_message": "Recipient name cannot be None",
        "error": "InvalidParameter",
        "payload": None
        }
    },
    "id": "request123"
}
```

Properti `payload` dalam output adalah respon dari Twilio API ketika pesan dikirim. Jika konektor mendeteksi bahwa data input tidak valid (sebagai contoh, itu tidak menentukan field input yang diperlukan), konektor mengembalikan kesalahan dan menetapkan nilai ke `None`. Berikut ini adalah contoh muatan:

```
{
    'from_number':'+19999999999',
    'messages': {
        'name':'Darla',
        'to_number':'+12345000000',
        'message_status':'undelivered'
    }
}
```

```
{
    'from_number':'+19999999999',
    'messages': {
        'name':'Darla',
        'to_number':'+12345000000',
        'message_status':'queued'
    }
}
```

Contoh Penggunaan

Gunakan langkah-langkah tingkat tinggi berikut untuk mengatur contoh fungsi Lambda Python 3.7 yang dapat Anda gunakan untuk mencoba konektor.

> **Note**
>
> Yangthe section called “Memulai dengan konektor (konsol)”danthe section called “Memulai dengan konektor (CLI)”Topik end-to-end langkah-langkah yang menunjukkan cara mengatur, dan menguji konektor Notifikasi Twilio

1. Pastikan Anda memenuhi persyaratan untuk konektor.
2. Buat dan terbitkan fungsi Lambda yang mengirimkan data input ke konektor.

   Simpan kode contoh sebagai file PY. Unduh dan unzip AWS IoT Greengrass Core SDK for Python. Kemudian, buat paket zip yang berisi file PY dan folder `greengrasssdk` dalam tingkat root. Paket zip ini adalah paket deployment yang Anda unggah ke AWS Lambda.

   Setelah Anda membuat fungsi Lambda Python 3.7, terbitkankan versi fungsi dan buat alias.
3. Konfigurasikan grup Greengrass Anda.

a. Tambahkan fungsi Lambda dengan aliasnya (direkomendasikan). Konfigurasikan siklus hidup Lambda sebagai berumur panjang (atau `"Pinned": true` dalam CLI).

b. Tambahkan sumber daya rahasia yang diperlukan dan berikan akses baca ke fungsi Lambda.

c. Tambahkan konektor dan konfigurasikan parameter.

d. Tambahkan langganan yang mengizinkan konektor untuk menerima data input dan mengirim data output pada filter topik yang didukung.

   - Atur fungsi Lambda sebagai sumber, konektor sebagai target, dan gunakan filter topik input yang mendukung.
   - Atur konektor sebagai sumber, AWS IoT Core sebagai target, dan gunakan filter topik input yang mendukung. Anda menggunakan langganan ini untuk melihat pesan status dalam konsol AWS IoT tersebut.

4. Men-deploy grup.

5. Di konsol AWS IoT tersebut, pada halaman Tes ini, berlangganan ke topik data output untuk melihat pesan status dari konektor. Contoh fungsi Lambda yang berumur panjang dan mulai mengirim pesan segera setelah grup dalam-deploy.

   Setelah selesai pengujian, Anda dapat mengatur siklus hidup Lambda ke sesuai permintaan (atau `"Pinned": false` dalam CLI) dan men-deploy grup. Ini menghentikan fungsi dari mengirim pesan.

Contoh

Contoh fungsi Lambda berikut mengirimkan pesan input ke konektor. Contoh ini memicu pesan teks.

```
import greengrasssdk
import json

iot_client = greengrasssdk.client('iot-data')
TXT_INPUT_TOPIC = 'twilio/txt'
CALL_INPUT_TOPIC = 'twilio/call'

def publish_basic_message():

    txt = {
        "request": {
```

```
            "recipient" : {
                "name": "Darla",
                "phone_number": "+12345000000",
                "message": 'Hello from the edge'
            },
            "from_number" : "+19999999999"
        },
        "id" : "request123"
    }

    print("Message To Publish: ", txt)

    client.publish(topic=TXT_INPUT_TOPIC,
                   payload=json.dumps(txt))

publish_basic_message()

def lambda_handler(event, context):
    return
```

Lisensi

Konektor Notifikasi Twilio mencakup perangkat lunak/lisensi pihak ketiga berikut ini:

- twilio-python/MIT

Konektor ini dirilis di bawah Perjanjian Lisensi Perangkat lunak core Greengrass.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi konektor.

Versi	Perubahan
5	Ditambahkan parameter `IsolationMode` untuk mengonfigurasi mode kontainerisasi untuk konektor.
4	Memperbarui waktu aktif Lambda untuk Python 3.7, yang mengubah persyaratan waktu aktif.

Versi	Perubahan
3	Perbaiki untuk mengurangi pencatatan berlebihan.
2	Penyempurnaan dan perbaikan bug kecil.
1	Pelepasan .

Sebuah grup Greengrass dapat berisi hanya satu versi konektor pada suatu waktu. Untuk informasi lebih lanjut tentang pembaruan versi konektor, lihat the section called “Versi upgrade konektor”.

Lihat juga

- Integrasikan dengan layanan dan protokol menggunakan konektor
- the section called “Memulai dengan konektor (konsol)”
- the section called “Memulai dengan konektor (CLI)”
- Referensi Twilio API

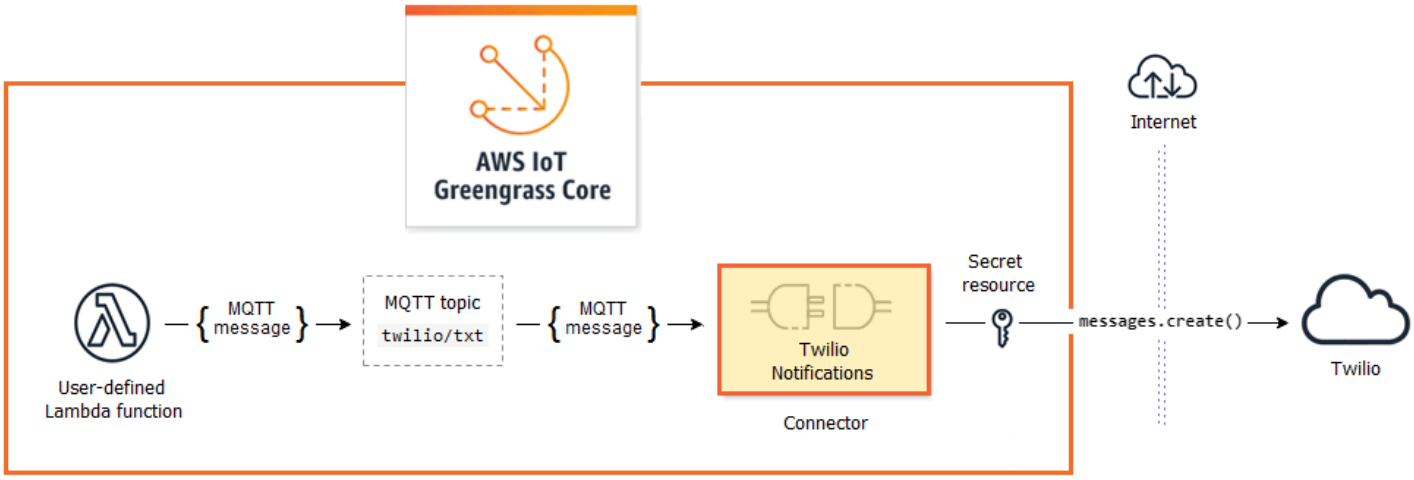
Memulai dengan konektor Greengrass (konsol)

Fitur ini tersedia untuk AWS IoT Greengrass Core v1.7 dan yang lebih baru.

Tutorial ini menunjukkan cara menggunakan AWS Management Console untuk bekerja dengan konektor.

Gunakan konektor untuk mempercepat siklus hidup pengembangan Anda. Konektor prebuilt, modul dapat digunakan kembali yang dapat membuatnya lebih mudah untuk berinteraksi dengan layanan, protokol, dan sumber daya. Mereka dapat membantu Anda men-deploy logika bisnis ke perangkat Greengrass lebih cepat. Untuk informasi lebih lanjut, lihat Integrasikan dengan layanan dan protokol menggunakan konektor.

Dalam tutorial ini, Anda mengkonfigurasi dan men-deploy konektor Notifikasi Twilio. Konektor menerima informasi pesan Twilio sebagai input data, dan kemudian memicu pesan teks Twilio. Aliran data ditunjukkan pada diagram berikut.



Setelah Anda mengkonfigurasi konektor, Anda membuat fungsi Lambda dan langganan.

- Fungsi mengevaluasi data simulasi dari sensor temperatur. Ini kondisional menerbitkan informasi pesan Twilio untuk topik MQTT. Ini adalah topik yang konektor berlangganan.
- Langganan membolehkan fungsi untuk menerbitkan topik dan konektor untuk menerima data dari topik tersebut.

Konektor Notifikasi Twilio memerlukan token Twilio auth untuk berinteraksi dengan Twilio API. Token adalah teks tipe rahasia yang dibuat di AWS Secrets Manager dan direferensikan dari sumber daya grup. Hal ini memungkinkan AWS IoT Greengrass untuk membuat salinan lokal rahasia pada Greengrass core, di mana itu dienkripsi dan tersedia untuk konektor. Untuk informasi lebih lanjut, lihat Men-deploy rahasia ke core.

Tutorial berisi langkah-langkah tingkat tinggi berikut:

1. Buat rahasia Secrets Manager
2. Tambahkan sumber daya rahasia ke grup
3. Tambahkan konektor ke grup
4. Buat paket deployment fungsi Lambda
5. Buat fungsi Lambda
6. Tambahkan fungsi ke grup
7. Menambahkan langganan ke grup
8. Men-deploy grup

9. the section called “Pengujian solusi”

Tutorial akan memakan waktu sekitar 20 menit untuk menyelesaikannya.

Prasyarat

Untuk menyelesaikan tutorial ini, Anda memerlukan:

- Sebuah grup Greengrass dan Greengrass core (v1.9.3 atau yang lebih baru). Untuk mempelajari cara membuat grup Greengrass dan core, lihat Memulai dengan AWS IoT Greengrass. Tutorial Memulai juga mencakup langkah-langkah untuk menginstal AWS IoT Greengrass perangkat lunak Core.
- Python 3.7 diinstal pada AWS IoT Greengrass perangkat core.
- AWS IoT Greengrass harus dikonfigurasi untuk mendukung rahasia lokal, seperti yang dijelaskan dalam Persyaratan Rahasia.

> **Note**
>
> Persyaratan ini mencakup akses ke rahasia Secrets Manager Anda. Jika Anda menggunakan peran layanan default Greengrass, Greengrass memiliki izin untuk mendapatkan nilai-nilai rahasia dengan nama yang dimulai dengan Greengrass-.

- Akun Twilio SID, token auth, dan nomor telepon berkemampuan Twilio. Setelah Anda membuat proyek Twilio, nilai-nilai ini tersedia dalam dasbor proyek.

> **Note**
>
> Anda dapat menggunakan akun percobaan Twilio. Jika menggunakan akun uji coba, Anda harus menambahkan nomor telepon penerima selain Twilio ke daftar nomor telepon terverifikasi. Untuk informasi lebih lanjut, lihat Cara Bekerja dengan Akun Trial Twilio Gratis.

Langkah 1: Buat rahasia Secrets Manager

Pada langkah ini, Anda menggunakan AWS Secrets Manager konsol untuk membuat rahasia tipe teks untuk token Twilio auth Anda.

1. Masuk ke konsol AWS Secrets Manager tersebut.

   > **Note**
   >
   > Untuk informasi selengkapnya tentang proses ini, lihatLangkah 1: Membuat dan menyimpan rahasia Anda diAWS Secrets Managerdi dalamAWS Secrets ManagerPanduan Pengguna.

2. Pilih Simpan rahasia baru.
3. Di bawahPilih tipe rahasia, pilihJenis rahasia lainnya.
4. Bawah Tentukan pasangan kunci/nilai yang akan disimpan untuk rahasia ini, pada plaintext tab, masukkan token Twilio auth Anda. Hapus semua format JSON dan masukkan hanya nilai token.
5. tetapaws/secretsmanagerchoose untuk kunci enkripsi, lalu pilihSelanjutnya.

   > **Note**
   >
   > Anda tidak dikenakan biaya oleh AWS KMS jika Anda menggunakan kunci mengelola AWS default yang dibuat oleh Secrets Manager di akun Anda.

6. Untuk Nama rahasia, masukkan **`greengrass-TwilioAuthToken`**, dan pilih Selanjutnya.

   > **Note**
   >
   > Secara default, peran layanan Greengrass memungkinkan AWS IoT Greengrass untuk mendapatkan nilai rahasia dengan nama yang dimulai dengan Greengrass-. Untuk informasi lebih lanjut, lihat persyaratan rahasia.

7. Tutorial ini tidak memerlukan rotasi, jadi pilih nonaktifkan rotasi otomatis, lalu pilihSelanjutnya.
8. Pada halaman Tinjauan tersebut, tinjau pengaturan Anda, dan kemudian pilih Menyimpan.

   Selanjutnya, Anda membuat sumber daya rahasia dalam grup Greengrass Anda yang mereferensi rahasia.

Langkah 2: Tambahkan sumber daya rahasia ke grup Greengrass

Pada langkah ini, Anda tambahkan sumber daya rahasia ke grup Greengrass. Sumber daya ini adalah referensi rahasia yang Anda buat pada langkah sebelumnya.

1. DiAWS IoTpanel navigasi konsol, di bawahKelola, perluasPerangkat Greengrass, dan kemudian pilihGrup (V1).
2. Pilih grup yang ingin Anda tambahkan sumber daya rahasia.
3. Pada halaman konfigurasi grup, pilihSumber dayatab, lalu gulir ke bawah keRahasiaBagian. YangRahasiabagian menampilkan sumber daya rahasia milik grup. Anda dapat menambahkan, menyunting, dan menghapus sumber daya rahasia dari bagian ini.

   > **Note**
   >
   > Sebagai alternatif, konsol tersebut mengizinkan Anda membuat sumber rahasia dan rahasia saat Anda mengonfigurasi konektor atau fungsi Lambda. Anda dapat melakukan hal ini dari konektor halaman Mengonfigurasi parameter atau fungsi Lambda halaman Sumber Daya ini.

4. PilihTambahkandi bawahRahasiaBagian.
5. PadaTambahkan sumber daya rahasiahalaman, masukkan**`MyTwilioAuthToken`**untukNama sumber daya.
6. UntukRahasia, pilihGreengrassTwilioAuthToken.
7. DiPilih label (Opsional)Bagian AWSCURRENT label pementasan mewakili versi terbaru dari rahasia. Label ini selalu disertakan dalam sumber rahasia.

   > **Note**
   >
   > Tutorial ini membutuhkan AWSCURRENT label saja. Anda dapat secara opsional menyertakan label yang diperlukan oleh fungsi Lambda atau konektor.

8. PilihTambahkan sumber daya.

Langkah 3: Tambahkan konektor ke grup Greengrass

Pada langkah ini, Anda mengkonfigurasi parameter untuk konektor Notifikasi Twilio dan menambahkannya ke grup.

1. Pada halaman konfigurasi grup, pilih Konektor, dan lalu pilih Menambahkan konektor.
2. PadaTambahkan konektorhalaman, pilihNotifikasi Twilio.
3. Pilih versi.

4. DiKonfigurasiBagian:

   - UntukSumber daya token autentikasi Twilio, masukkan sumber daya yang Anda buat di langkah sebelumnya.

     > **Note**
     >
     > Saat Anda memasukkan sumber daya,ARN dari Twilio auth token rahasiaproperti dihuni untuk Anda.

   - Untuk Default dari nomor telepon, masukkan nomor telepon berkemampuan Twilio Anda.
   - Untuk akun Twilio SID, masukkan akun Twilio SID Anda.

5. PilihTambahkan sumber daya.

Langkah 4: Buat paket deployment fungsi Lambda

Untuk membuat fungsi Lambda, Anda harus terlebih dahulu membuat fungsi Lambda paket deployment yang berisi kode fungsi dan dependensi. Fungsi Greengrass Lambda membutuhkan AWS IoT Greengrass Core SDK untuk tugas seperti berkomunikasi dengan pesan MQTT di lingkungan core dan mengakses rahasia lokal. Tutorial ini menciptakan fungsi Python, sehingga Anda menggunakan versi Python dari SDK dalam paket deployment.

1. Dari halaman unduh AWS IoT Greengrass Core SDK ini, unduh AWS IoT Greengrass Core SDK for Python ke komputer Anda.
2. Unzip paket yang diunduh untuk mendapatkan SDK. SDK adalah folder `greengrasssdk` tersebut.
3. Simpan fungsi kode Python berikut dalam sebuah file lokal bernama `temp_monitor.py`.

```
import greengrasssdk
import json
import random

client = greengrasssdk.client('iot-data')

# publish to the Twilio Notifications connector through the twilio/txt topic
def function_handler(event, context):
    temp = event['temperature']
```

```
    # check the temperature
    # if greater than 30C, send a notification
    if temp > 30:
        data = build_request(event)
        client.publish(topic='twilio/txt', payload=json.dumps(data))
        print('published:' + str(data))

    print('temperature:' + str(temp))
    return

# build the Twilio request from the input data
def build_request(event):
    to_name = event['to_name']
    to_number = event['to_number']
    temp_report = 'temperature:' + str(event['temperature'])

    return {
        "request": {
            "recipient": {
                "name": to_name,
                "phone_number": to_number,
                "message": temp_report
            }
        },
        "id": "request_" + str(random.randint(1,101))
    }
```

4. Zip item berikut ke dalam file bernama `temp_monitor_python.zip`. Saat membuat file ZIP, sertakan hanya kode dan dependensi, bukan folder yang berisi.

   - temp_monitor.py. Aplikasi logic.
   - greengrasssdk. Diperlukan perpustakaan untuk fungsi Python Greengrass Lambda yang menerbitkan pesan MQTT.

   Ini adalah paket deployment fungsi Lambda Anda.

Sekarang, buat fungsi Lambda yang menggunakan paket deployment.

Langkah 5: Membuat fungsi Lambda diAWS Lambdakonsol

Pada langkah ini, Anda menggunakan konsol AWS Lambda untuk membuat fungsi Lambda dan mengonfigurasinya agar menggunakan paket deployment Anda. Kemudian, Anda mempublikasikan versi fungsi dan membuat alias.

1. Pertama, buat fungsi Lambda.

   a. Di AWS Management Console, pilih Layanan, dan buka konsol AWS Lambda tersebut.

   b. Pilih Buat fungsi dan kemudian Tulis dari awal.

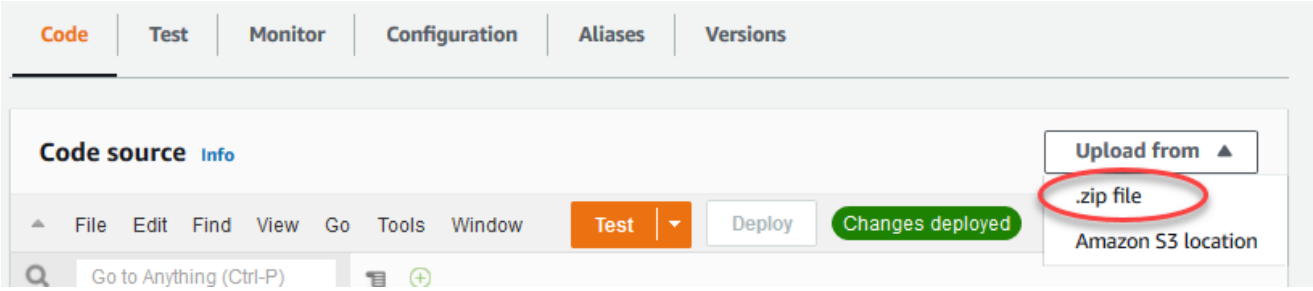
   c. Di bagian Informasi dasar tersebut, gunakan nilai-nilai berikut:

      - Untuk Nama fungsi, masukkan **`TempMonitor`**.
      - Untuk Waktu pengoperasian, pilih Python 3.7.
      - Untuk Izin, pertahankan pengaturan default. Hal ini menciptakan peran eksekusi yang memberikan izin Lambda basic. Peran ini tidak digunakan oleh AWS IoT Greengrass.

   d. Di bagian bawah halaman, pilih Buat Fungsi.

2. Selanjutnya, daftarkan handler dan unggah paket deployment fungsi Lambda Anda.

   a. Pada tab Kode ini, di bawah Sumber kode, pilih Unggah dari. Dari dropdown, pilih file .zip.



   b. Pilih Unggah, lalu pilih paket deployment `temp_monitor_python.zip` Anda. Lalu, pilih Simpan.

   c. Pada tab Kode fungsi, di bawah Pengaturan waktu aktif, pilih Edit, dan kemudian masukkan nilai-nilai berikut.

      - Untuk Waktu pengoperasian, pilih Python 3.7.
      - Untuk Handler, masukkan **`temp_monitor.function_handler`**

   d. Pilih Save (Simpan).

> **Note**
>
> Tombol Tes pada konsol AWS Lambda tidak bekerja dengan fungsi ini. Pada AWS IoT Greengrass Core SDK tidak berisi modul yang diperlukan untuk menjalankan fungsi Greengrass Lambda Anda secara independen di konsol AWS Lambda tersebut. Modul-modul ini (misalnya, `greengrass_common`) dipasok ke fungsi setelah mereka di-deploy ke core Greengrass Anda.

3. Sekarang, publikasikan versi pertama fungsi Lambda Anda dan membuat alias untuk versi.

> **Note**
>
> Grup Greengrass dapat mereferensi fungsi Lambda dengan alias (direkomendasikan) atau dengan versi. Menggunakan alias membuatnya lebih mudah untuk mengelola pembaruan kode karena Anda tidak perlu mengubah tabel langganan atau definisi grup ketika kode fungsi diperbarui. Sebaliknya, Anda hanya mengarahkan alias ke versi fungsi baru.

   a. Dari menu Tindakan ini, pilih Terbitkan versi baru.
   b. Untuk Versi Deskripsi, masukkan **`First version`**, lalu pilih Publikasikan.
   c. PadaTempMonitor: 1halaman konfigurasi, dariTindakanmenu, pilihMembuat alias.
   d. Pada halaman Buat alias baru ini, gunakan nilai-nilai berikut:

      - Untuk Nama, masukkan **`GG_TempMonitor`**.
      - Untuk Versi, pilih 1.

      > **Note**
      >
      > AWS IoT Greengrass tidak support alias Lambda untuk versi $TERBARU ini.

   e. Pilih Create (Buat).

Sekarang Anda siap untuk menambahkan fungsi Lambda ke grup Greengrass Anda.

Langkah 6: Tambahkan fungsi Lambda ke grup Greengrass

Dalam langkah ini, Anda menambahkan fungsi Lambda ke grup lalu mengonfigurasi siklus hidup dan lingkungan variabel. Untuk informasi selengkapnya, lihat the section called “Mengontrol eksekusi fungsi Greengrass Lambda”.

1. Pada halaman konfigurasi grup, pilihFungsi LambdaTab.
2. Di bawahFungsi Lambda, pilihTambahkan.
3. PadaTambahkan fungsi Lambdahalaman, pilihTempMonitoruntuk fungsi Lambda Anda.
4. UntukVersi fungsi Lambda, pilihAlias: GG_TempMonitor.
5. PilihTambahkan fungsi Lambda.

Langkah 7: Tambahkan langganan ke grup Greengrass

Pada langkah ini, Anda menambahkan langganan yang memungkinkan fungsi Lambda untuk mengirim input data ke konektor. Konektor mendefinisikan topik MQTT yang berlangganan, jadi langganan ini menggunakan salah satu topik. Ini adalah topik yang sama bahwa fungsi contoh menerbitkan.

Untuk tutorial ini, Anda juga membuat langganan yang memungkinkan fungsi untuk menerima pembacaan temperatur simulasi dari AWS IoT dan membolehkan AWS IoT untuk menerima informasi status dari konektor.

1. Pada halaman konfigurasi grup, pilihLangganantab, dan kemudian pilihTambahkan langganan.
2. PadaBuat langgananhalaman, konfigurasikan sumber dan target, sebagai berikut:
   a. UntukJenis Sumber, pilihFungsi Lambda, dan kemudian pilihTempMonitor.
   b. UntukJenis target, pilihKonektor, dan kemudian pilihNotifikasi Twilio.
3. UntukFilter topik, pilih**`twilio/txt`**.
4. Pilih Buat langganan.
5. Ulangi langkah 1 - 4 untuk membuat langganan yang memungkinkan AWS IoT untuk menerbitkan pesan ke fungsi.
   a. UntukJenis Sumber, pilihLayanan, dan kemudian pilihIoT Cloud.
   b. UntukPilih target, pilihFungsi Lambda, dan kemudian pilihTempMonitor.
   c. Untuk filter Topik, masukkan **`temperature/input`**.

6. Ulangi langkah 1 - 4 untuk membuat langganan yang membolehkan konektor menerbitkan pesan AWS IoT.

   a. UntukJenis Sumber, pilihKonektor, dan kemudian pilihNotifikasi Twilio.

   b. UntukJenis target, pilihLayanan, dan kemudian pilihIoT Cloud.

   c. Untuk filter Topik, **`twilio/message/status`** dimasukkan untuk Anda. Ini adalah topik yang telah ditetapkan yang diterbitkan oleh konektor.

Langkah 8: Men-deploy grup Greengrass

Men-deploy grup ke perangkat core.

1. Pastikan bahwa core AWS IoT Greengrass sedang berjalan. Jalankan perintah berikut di terminal Raspberry Pi Anda, sesuai kebutuhan.

   a. Untuk memeriksa apakah daemon sedang berjalan:

   ```
   ps aux | grep -E 'greengrass.*daemon'
   ```

   Jika outputnya berisi entri `root` untuk `/greengrass/ggc/packages/`*`ggc-version`*`/bin/daemon`, maka daemon sedang berjalan.

   > **Note**
   >
   > Versi di jalur tergantung pada versi perangkat lunak AWS IoT Greengrass core yang diinstal pada perangkat core Anda.

   b. Untuk memulai daemon:

   ```
   cd /greengrass/ggc/core/
   sudo ./greengrassd start
   ```

2. Pada halaman konfigurasi grup, pilihDeploy.

3. a. DiFungsi Lambdatab, di bawahFungsi Lambda sistembagian, pilihDetektor IPdan pilihlahedit.

   b. DiPengaturan detektor IPkotak dialog, pilihSecara otomatis mendeteksi dan mengganti titik akhir broker MQTT.

c. Pilih Save (Simpan).

Hal ini mengaktifkan perangkat untuk secara otomatis memperoleh informasi konektivitas untuk core, seperti alamat IP, DNS, dan nomor port. Deteksi otomatis direkomendasikan, namun AWS IoT Greengrass juga support titik akhir yang ditentukan secara manual. Anda hanya diminta untuk metode penemuan pertama kalinya bahwa grup di-deploy.

> **Note**
>
> Jika diminta, berikan izin untuk membuat Peran layanan Greengrass dan kaitkan dengan Akun AWS Anda pada Wilayah AWS. Peran ini memungkinkan AWS IoT Greengrass untuk mengakses sumber daya Anda di layanan AWS ini.

Halaman Deployment menampilkan timestamp deployment, ID versi, dan status. Setelah selesai, status yang ditampilkan untuk deployment harusCompleted (Lengkap).

Untuk langkah-langkah penyelesaian masalah, lihat Memecahkan masalah.

> **Note**
>
> Sebuah grup Greengrass dapat berisi hanya satu versi dari konektor pada suatu waktu. Untuk informasi tentang peningkatan versi konektor, lihat the section called “Versi upgrade konektor”.

Pengujian solusi

1. Pada halaman beranda konsol AWS IoT tersebut, pilih Pengujian.
2. Untuk Berlangganan topik, gunakan nilai berikut, dan lalu pilih Langganan. Konektor Notifikasi Twilio menerbitkan informasi status untuk topik ini.

Properti	Nilai
Topik langganan	twilio/message/status
Tampilan muatan MQTT	Tampilkan muatan sebagai string

3. Untuk Publikasikan ke topik, gunakan nilai-nilai berikut, dan kemudian pilih Publikasikan untuk memanggil fungsi.

Properti	Nilai
Topik	temperatur/input
Message	Ganti*nama-penerima* dengan nama dan*recipient-phone-number* dengan nomor telepon penerima pesan teks. Contoh: `+12345000000` `{ "to_name": " recipient-name ", "to_number": " recipient-phone-number ", "temperature": 31 }` Jika Anda menggunakan akun uji coba, Anda harus menambahkan nomor telepon penerima selain Twilio ke daftar nomor telepon terverifikasi. Untuk informasi lebih lanjut, lihat Verifikasi Nomor Telepon Pribadi.

Jika berhasil, penerima menerima pesan teks dan konsol menampilkan `success` status dari output data.

Sekarang, ubah `temperature` dalam pesan input untuk **29** dan terbitkan. Karena ini adalah kurang dari 30, TempMonitor Fungsi tidak memicu pesan Twilio.

Lihat juga

- Integrasikan dengan layanan dan protokol menggunakan konektor
- the section called “AWS-disediakan konektor Greengrass”

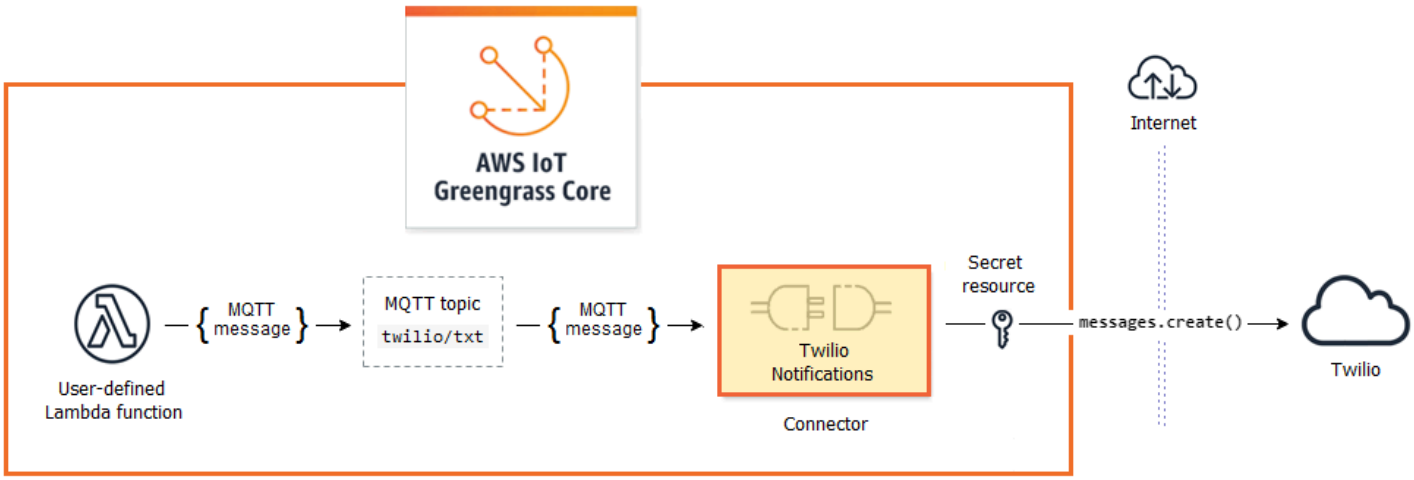
Memulai dengan konektor Greengrass (CLI)

Fitur ini tersedia untuk AWS IoT Greengrass Core v1.7 dan yang lebih baru.

Tutorial ini menunjukkan cara menggunakan AWS CLI untuk bekerja dengan konektor.

Gunakan konektor untuk mempercepat siklus hidup pengembangan Anda. Konektor prebuilt, modul dapat digunakan kembali yang dapat membuatnya lebih mudah untuk berinteraksi dengan layanan, protokol, dan sumber daya. Mereka dapat membantu Anda men-deploy logika bisnis ke perangkat Greengrass lebih cepat. Untuk informasi lebih lanjut, lihat Integrasikan dengan layanan dan protokol menggunakan konektor.

Dalam tutorial ini, Anda mengkonfigurasi dan men-deploy konektor Notifikasi Twilio. Konektor menerima informasi pesan Twilio sebagai input data, dan kemudian memicu pesan teks Twilio. Aliran data ditunjukkan pada diagram berikut.



Setelah Anda mengkonfigurasi konektor, Anda membuat fungsi Lambda dan langganan.

- Fungsi mengevaluasi data simulasi dari sensor temperatur. Ini kondisional menerbitkan informasi pesan Twilio untuk topik MQTT. Ini adalah topik yang konektor berlangganan.
- Langganan membolehkan fungsi untuk menerbitkan topik dan konektor untuk menerima data dari topik tersebut.

Konektor Notifikasi Twilio memerlukan token Twilio auth untuk berinteraksi dengan Twilio API. Token adalah teks tipe rahasia yang dibuat di AWS Secrets Manager dan direferensikan dari sumber daya grup. Hal ini memungkinkan AWS IoT Greengrass untuk membuat salinan lokal rahasia pada

Greengrass core, di mana itu dienkripsi dan tersedia untuk konektor. Untuk informasi lebih lanjut, lihat Men-deploy rahasia ke core.

Tutorial berisi langkah-langkah tingkat tinggi berikut:

1. Buat rahasia Secrets Manager
2. Buat definisi sumber daya dan versi
3. Buat definisi konektor dan versi
4. Buat paket deployment fungsi Lambda
5. Buat fungsi Lambda
6. Buat definisi fungsi dan versi
7. Buat definisi langganan dan versi
8. Buat versi grup
9. Buat deployment
10. the section called “Uji solusi”

Tutorial harus memerlukan waktu sekitar 30 menit untuk menyelesaikannya.

MenggunakanAWS IoT GreengrassAPI

Hal itu sangat membantu untuk memahami pola berikut ketika Anda bekerja dengan grup Greengrass dan komponen grup (misalnya, konektor, fungsi, dan sumber daya dalam grup).

- Di bagian atas hirarki, komponen memiliki definisi objek yang merupakan wadah untuk versi objek. Sebaliknya, versi adalah kotak untuk penyambung, fungsi, atau jenis komponen lain.
- Ketika Anda menerapkan ke Greengrass core, Anda menerapkan versi grup tertentu. Versi grup dapat berisi satu versi dari setiap jenis komponen. Core diperlukan, tetapi yang lain disertakan sesuai kebutuhan.
- Versi tidak berubah, jadi Anda harus membuat versi baru ketika Anda ingin membuat perubahan.

> **Tip**
>
> Jika Anda menerima kesalahan ketika Anda menjalankan AWS CLI perintah, tambahkan parameter `--debug` dan jalankan kembali perintah tersebut untuk mendapatkan informasi lebih lanjut tentang kesalahan tersebut.

API AWS IoT Greengrass memungkinkan Anda membuat beberapa definisi untuk jenis komponen. Sebagai contoh, Anda dapat membuat `FunctionDefinition` objek setiap kali Anda membuat `FunctionDefinitionVersion`, atau Anda dapat tambahkan versi baru ke definisi yang ada. Fleksibilitas ini membolehkan Anda untuk menyesuaikan sistem manajemen versi Anda.

Prasyarat

Untuk menyelesaikan tutorial ini, Anda memerlukan:

- Sebuah grup Greengrass dan Greengrass core (v1.9.3 atau yang lebih baru). Untuk mempelajari cara membuat grup Greengrass dan core, lihat Memulai dengan AWS IoT Greengrass. Tutorial Memulai juga mencakup langkah-langkah untuk menginstal AWS IoT Greengrass perangkat lunak Core.
- Python 3.7 diinstal pada AWS IoT Greengrass perangkat core.
- AWS IoT Greengrass harus dikonfigurasi untuk mendukung rahasia lokal, seperti yang dijelaskan dalam Persyaratan Rahasia.

  > **Note**
  >
  > Persyaratan ini mencakup akses ke rahasia Secrets Manager Anda. Jika Anda menggunakan peran layanan default Greengrass, Greengrass memiliki izin untuk mendapatkan nilai-nilai rahasia dengan nama yang dimulai dengan Greengrass-.

- Akun Twilio SID, token auth, dan nomor telepon berkemampuan Twilio. Setelah Anda membuat proyek Twilio, nilai-nilai ini tersedia dalam dasbor proyek.

  > **Note**
  >
  > Anda dapat menggunakan akun percobaan Twilio. Jika menggunakan akun uji coba, Anda harus menambahkan nomor telepon penerima selain Twilio ke daftar nomor telepon terverifikasi. Untuk informasi lebih lanjut, lihat Cara Bekerja dengan Akun Trial Twilio Gratis.

- AWS CLI diinstal dan dikonfigurasi pada komputer Anda. Untuk informasi lebih lanjut, lihat Menginstal AWS Command Line Interface dan Mengonfigurasi AWS CLI di AWS Command Line Interface Panduan Pengguna.

Contoh dalam tutorial ini ditulis untuk Linux dan sistem berbasis Unix lainnya. Jika Anda menggunakan Windows, lihat Menentukan nilai parameter untuk AWS Command Line Interface untuk memelajari perbedaan sintaks.

Jika perintah berisi string JSON, tutorial memberikan contoh yang memiliki JSON pada satu baris. Pada beberapa sistem, mungkin lebih mudah untuk mengedit dan menjalankan perintah menggunakan format ini.

Langkah 1: Buat rahasia Secrets Manager

Pada langkah ini, Anda menggunakan opsi AWS Secrets Manager API untuk membuat rahasia untuk token auth Twilio Anda.

1. Pertama, buat rahasia.

   - Ganti*twilio-auth-token*dengan token auth Twilio Anda.

   ```
   aws secretsmanager create-secret --name greengrass-TwilioAuthToken --secret-
   string twilio-auth-token
   ```

   > **Note**
   >
   > Secara default, peran layanan Greengrass memungkinkan AWS IoT Greengrass untuk mendapatkan nilai rahasia dengan nama yang dimulai dengan Greengrass-. Untuk informasi lebih lanjut, lihat rahasia persyaratan.

2. Salin ARN rahasia dari output. Anda menggunakan ini untuk membuat sumber daya rahasia dan mengkonfigurasi konektor Notifikasi Twilio.

Langkah 2: Buat definisi sumber daya dan versi

Pada langkah ini, Anda menggunakan AWS IoT Greengrass API untuk membuat sumber daya rahasia untuk rahasia Secrets Manager Anda.

1. Buat definisi sumber daya yang mencakup versi awal.

- Ganti *rahasia-arn* dengan ARN dari rahasia yang Anda salin pada langkah sebelumnya.

JSON Expanded

```
aws greengrass create-resource-definition --name MyGreengrassResources --
initial-version '{
    "Resources": [
        {
            "Id": "TwilioAuthToken",
            "Name": "MyTwilioAuthToken",
            "ResourceDataContainer": {
                "SecretsManagerSecretResourceData": {
                    "ARN": "secret-arn"
                }
            }
        }
    ]
}'
```

JSON Single-line

```
aws greengrass create-resource-definition \
--name MyGreengrassResources \
--initial-version '{"Resources": [{"Id": "TwilioAuthToken",
 "Name": "MyTwilioAuthToken", "ResourceDataContainer":
 {"SecretsManagerSecretResourceData": {"ARN": "secret-arn"}}}]}'
```

2. Salin `LatestVersionArn` definisi sumber daya dari output. Anda menggunakan nilai ini untuk menambahkan versi definisi sumber daya untuk versi grup yang Anda terapkan ke core.

Langkah 3: Buat definisi konektor dan versi

Dalam langkah ini, Anda mengkonfigurasi parameter untuk konektor Pemberitahuan Twilio.

1. Buat definisi konektor dengan versi awal.

   - Ganti *akun-sid* dengan akun Twilio SID Anda.

- Ganti *rahasia-arn* dengan ARN rahasia Secrets Manager Anda. Konektor menggunakan ini untuk mendapatkan nilai dari rahasia lokal.
- Ganti *nomor telepon* dengan nomor telepon berkemampuan Twilio Anda. Twilio menggunakan ini untuk memulai pesan teks. Hal ini dapat diganti dalam muatan pesan masukan. Gunakan format berikut: +19999999999.

JSON Expanded

```
aws greengrass create-connector-definition --name MyGreengrassConnectors --
initial-version '{
    "Connectors": [
        {
            "Id": "MyTwilioNotificationsConnector",
            "ConnectorArn": "arn:aws:greengrass:region::/connectors/
TwilioNotifications/versions/4",
            "Parameters": {
                "TWILIO_ACCOUNT_SID": "account-sid",
                "TwilioAuthTokenSecretArn": "secret-arn",
                "TwilioAuthTokenSecretArn-ResourceId": "TwilioAuthToken",
                "DefaultFromPhoneNumber": "phone-number"
            }
        }
    ]
}'
```

JSON Single-line

```
aws greengrass create-connector-definition \
--name MyGreengrassConnectors \
--initial-version '{"Connectors": [{"Id": "MyTwilioNotificationsConnector",
 "ConnectorArn": "arn:aws:greengrass:region::/connectors/TwilioNotifications/
versions/4", "Parameters": {"TWILIO_ACCOUNT_SID": "account-sid",
 "TwilioAuthTokenSecretArn": "secret-arn", "TwilioAuthTokenSecretArn-
ResourceId": "TwilioAuthToken", "DefaultFromPhoneNumber": "phone-number"}}]}'
```

> **Note**
>
> `TwilioAuthToken` adalah ID yang Anda gunakan dalam langkah sebelumnya untuk membuat sumber daya rahasia.

2. Salin `LatestVersionArn` definisi konektor dari output. Anda menggunakan nilai ini untuk menambahkan versi definisi konektor ke versi grup yang Anda terapkan ke core.

Langkah 4: Buat paket deployment fungsi Lambda

Untuk membuat fungsi Lambda, Anda harus terlebih dahulu membuat fungsi Lambda paket deployment yang berisi kode fungsi dan dependensi. Fungsi Greengrass Lambda membutuhkan AWS IoT Greengrass Core SDK untuk tugas seperti berkomunikasi dengan pesan MQTT di lingkungan core dan mengakses rahasia lokal. Tutorial ini menciptakan fungsi Python, sehingga Anda menggunakan versi Python dari SDK dalam paket deployment.

1. Dari halaman unduh AWS IoT Greengrass Core SDK ini, unduh AWS IoT Greengrass Core SDK for Python ke komputer Anda.
2. Unzip paket yang diunduh untuk mendapatkan SDK. SDK adalah folder `greengrasssdk` tersebut.
3. Simpan fungsi kode Python berikut dalam sebuah file lokal bernama `temp_monitor.py`.

```
import greengrasssdk
import json
import random

client = greengrasssdk.client('iot-data')

# publish to the Twilio Notifications connector through the twilio/txt topic
def function_handler(event, context):
    temp = event['temperature']

    # check the temperature
    # if greater than 30C, send a notification
    if temp > 30:
        data = build_request(event)
        client.publish(topic='twilio/txt', payload=json.dumps(data))
        print('published:' + str(data))
```

```
    print('temperature:' + str(temp))
    return

# build the Twilio request from the input data
def build_request(event):
    to_name = event['to_name']
    to_number = event['to_number']
    temp_report = 'temperature:' + str(event['temperature'])

    return {
        "request": {
            "recipient": {
                "name": to_name,
                "phone_number": to_number,
                "message": temp_report
            }
        },
        "id": "request_" + str(random.randint(1,101))
    }
```

4. Zip item berikut ke dalam file bernama `temp_monitor_python.zip`. Saat membuat file ZIP, sertakan hanya kode dan dependensi, bukan folder yang berisi.

   - temp_monitor.py. Aplikasi logic.
   - greengrasssdk. Diperlukan perpustakaan untuk fungsi Python Greengrass Lambda yang menerbitkan pesan MQTT.

   Ini adalah paket deployment fungsi Lambda Anda.

Langkah 5: Buat fungsi Lambda

Sekarang, buat fungsi Lambda yang menggunakan paket deployment.

1. Buat IAM role sehingga Anda dapat lulus dalam peran ARN ketika Anda membuat fungsi.

   JSON Expanded

   ```
   aws iam create-role --role-name Lambda_empty --assume-role-policy '{
       "Version": "2012-10-17",
       "Statement": [
   ```

```
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "lambda.amazonaws.com"
            },
           "Action": "sts:AssumeRole"
        }
    ]
}'
```

JSON Single-line

```
aws iam create-role --role-name Lambda_empty --assume-role-policy '{"Version":
 "2012-10-17", "Statement": [{"Effect": "Allow", "Principal": {"Service":
 "lambda.amazonaws.com"},"Action": "sts:AssumeRole"}]}'
```

> **Note**
>
> AWS IoT Greengrass tidak menggunakan peran ini karena izin untuk fungsi Lambda Greengrass ditentukan di dalam peran grup Greengrass. Untuk tutorial ini, Anda membuat peran kosong.

2. Salin `Arn` dari output.
3. MenggunakanAWS LambdaAPI untuk membuat TempMonitor fungsi. Perintah berikut mengasumsikan bahwa file zip dalam direktori saat ini.

   - Ganti *`role-arn`* dengan `Arn` yang Anda salin.

```
aws lambda create-function \
--function-name TempMonitor \
--zip-file fileb://temp_monitor_python.zip \
--role role-arn \
--handler temp_monitor.function_handler \
--runtime python3.7
```

4. Mempublikasikan versi fungsi.

```
aws lambda publish-version --function-name TempMonitor --description 'First
 version'
```

5. Buat alias untuk versi yang dipublikasikan.

   Grup Greengrass dapat mereferensi fungsi Lambda dengan alias (direkomendasikan) atau dengan versi. Menggunakan alias membuatnya lebih mudah untuk mengelola pembaruan kode karena Anda tidak perlu mengubah tabel langganan atau definisi grup ketika kode fungsi diperbarui. Sebaliknya, Anda hanya mengarahkan alias ke versi fungsi baru.

   > **Note**
   >
   > AWS IoT Greengrass tidak mendukung alias Lambda untuk versi $TERBARU ini.

   ```
   aws lambda create-alias --function-name TempMonitor --name GG_TempMonitor --
   function-version 1
   ```

6. Salin `AliasArn` dari output. Anda menggunakan nilai ini ketika Anda mengkonfigurasi fungsi untuk AWS IoT Greengrass dan saat Anda membuat langganan.

Sekarang Anda siap untuk mengkonfigurasi fungsi untuk AWS IoT Greengrass.

Langkah 6: Buat definisi fungsi dan versi

Untuk menggunakan fungsi Lambda pada AWS IoT Greengrass inti, Anda buat versi definisi fungsi yang referensi fungsi Lambda dengan alias dan mendefinisikan konfigurasi grup-tingkat. Untuk informasi lebih lanjut, lihat the section called “Mengontrol eksekusi fungsi Greengrass Lambda”.

1. Buat definisi fungsi yang mencakup versi awal.

   - Ganti *`Alias-arn`* dengan `AliasArn` yang Anda salin ketika membuat alias.

JSON Expanded

```
aws greengrass create-function-definition --name MyGreengrassFunctions --
initial-version '{
    "Functions": [
        {
            "Id": "TempMonitorFunction",
            "FunctionArn": "alias-arn",
            "FunctionConfiguration": {
                "Executable": "temp_monitor.function_handler",
                "MemorySize": 16000,
                "Timeout": 5
            }
        }
    ]
}'
```

JSON Single-line

```
aws greengrass create-function-definition \
--name MyGreengrassFunctions \
--initial-version '{"Functions": [{"Id": "TempMonitorFunction",
 "FunctionArn": "alias-arn", "FunctionConfiguration": {"Executable":
 "temp_monitor.function_handler", "MemorySize": 16000,"Timeout": 5}}]}'
```

2. Salin `LatestVersionArn` dari output. Anda menggunakan nilai ini untuk menambahkan versi definisi fungsi untuk versi grup yang Anda terapkan ke inti.
3. Salin `Id` dari output. Anda menggunakan nilai ini kemudian ketika Anda memperbarui fungsi.

Langkah 7: Buat definisi langganan dan versi

Pada langkah ini, Anda tambahkan langganan yang memungkinkan fungsi Lambda untuk mengirim input data ke konektor. Konektor mendefinisikan topik MQTT yang berlangganan, jadi langganan ini menggunakan salah satu topik. Ini adalah topik yang sama bahwa fungsi contoh menerbitkan.

Untuk tutorial ini, Anda juga buat langganan yang memungkinkan fungsi untuk menerima pembacaan suhu simulasi dari AWS IoT dan izinkan AWS IoT untuk menerima informasi status dari konektor.

1. Buat definisi langganan yang berisi versi awal yang mencakup langganan.

- Ganti *Alias-arn* dengan `AliasArn` yang Anda salin ketika Anda membuat alias untuk fungsi. Gunakan ARN ini untuk kedua langganan yang menggunakannya.

JSON Expanded

```
aws greengrass create-subscription-definition --initial-version '{
    "Subscriptions": [
        {
            "Id": "TriggerNotification",
            "Source": "alias-arn",
            "Subject": "twilio/txt",
            "Target": "arn:aws:greengrass:region::/connectors/
TwilioNotifications/versions/4"
        },
        {
            "Id": "TemperatureInput",
            "Source": "cloud",
            "Subject": "temperature/input",
            "Target": "alias-arn"
        },
        {
            "Id": "OutputStatus",
            "Source": "arn:aws:greengrass:region::/connectors/
TwilioNotifications/versions/4",
            "Subject": "twilio/message/status",
            "Target": "cloud"
        }
    ]
}'
```

JSON Single-line

```
aws greengrass create-subscription-definition \
--initial-version '{"Subscriptions": [{"Id": "TriggerNotification", "Source":
 "alias-arn", "Subject": "twilio/txt", "Target": "arn:aws:greengrass:region::/
connectors/TwilioNotifications/versions/4"},{"Id": "TemperatureInput",
 "Source": "cloud", "Subject": "temperature/input", "Target": "alias-arn"},
{"Id": "OutputStatus", "Source": "arn:aws:greengrass:region::/connectors/
```

```
TwilioNotifications/versions/4", "Subject": "twilio/message/status", "Target":
 "cloud"}]}'
```

2. Salin `LatestVersionArn` dari output. Anda gunakan nilai ini untuk menambahkan versi definisi langganan ke versi grup yang Anda terapkan ke core.

Langkah 8: Buat versi grup

Sekarang, Anda siap untuk membuat versi grup yang berisi semua item yang ingin Anda terapkan. Anda melakukannya dengan membuat versi grup yang referensi versi target dari setiap jenis komponen.

Pertama, dapatkan ID grup dan ARN versi definisi core. Nilai-nilai ini diperlukan untuk membuat versi grup.

1. Dapatkan ID grup dan versi grup terbaru:

   a. Dapatkan ID dari grup Greengrass target dan versi grup. Prosedur ini mengasumsikan bahwa ini adalah versi grup dan grup terbaru. Query berikut mengembalikan grup yang paling baru dibuat.

   ```
   aws greengrass list-groups --query "reverse(sort_by(Groups,
    &CreationTimestamp))[0]"
   ```

   Atau, Anda dapat melakukan query berdasarkan nama. Nama grup tidak perlu unik, sehingga beberapa grup mungkin ditampilkan.

   ```
   aws greengrass list-groups --query "Groups[?Name=='MyGroup']"
   ```

   > **Note**
   >
   > Anda juga dapat menemukan nilai-nilai ini di konsol AWS IoT tersebut. ID grup ditampilkan pada halaman Pengaturan grup. ID versi grup ditampilkan pada grupDeploymenttab.

   b. Salin `Id` dari grup target dari output. Anda menggunakan ini untuk mendapatkan versi definisi core dan ketika Anda men-deploy grup.

c. Salin `LatestVersion` dari output, yang merupakan ID dari versi terakhir ditambahkan ke grup. Anda menggunakan ini untuk mendapatkan versi definisi core.

2. Dapatkan ARN dari versi definisi core:

   a. Dapatkan versi grup. Untuk langkah ini, kita berasumsi bahwa versi grup terbaru mencakup versi definisi core.

      - Ganti *`grup-id`* dengan `Id` yang Anda salin untuk grup.
      - Ganti*`group-version-id`*dengan`LatestVersion`yang Anda salin untuk grup.

```
aws greengrass get-group-version \
--group-id group-id \
--group-version-id group-version-id
```

   b. Salin `CoreDefinitionVersionArn` dari output.

3. Buat versi grup.

   - Ganti *`grup-id`* dengan `Id` yang Anda salin untuk grup.
   - Ganti*`core-definition-version-arn`*dengan`CoreDefinitionVersionArn`yang Anda salin untuk versi definisi core.
   - Ganti*`resource-definition-version-arn`*dengan`LatestVersionArn`yang Anda salin untuk definisi sumber daya.
   - Ganti*`connector-definition-version-arn`*dengan`LatestVersionArn`yang Anda salin untuk definisi konektor.
   - Ganti*`function-definition-version-arn`*dengan`LatestVersionArn`yang Anda salin untuk definisi fungsi.
   - Ganti*`subscription-definition-version-arn`*dengan`LatestVersionArn`yang Anda salin untuk definisi langganan.

```
aws greengrass create-group-version \
--group-id group-id \
--core-definition-version-arn core-definition-version-arn \
--resource-definition-version-arn resource-definition-version-arn \
--connector-definition-version-arn connector-definition-version-arn \
--function-definition-version-arn function-definition-version-arn \
--subscription-definition-version-arn subscription-definition-version-arn
```

4. Salin nilai `Version` dari outputnya. Ini adalah ID dari versi grup. Anda menggunakan nilai ini untuk menerapkan versi grup.

Langkah 9: Buat deployment

Terapkan grup ke perangkat core.

1. Di terminal perangkat core, pastikan bahwa AWS IoT Greengrass daemon sedang berjalan.

   a. Untuk memeriksa apakah daemon sedang berjalan:

   ```
   ps aux | grep -E 'greengrass.*daemon'
   ```

   Jika output berisi `root` entri untuk `/greengrass/ggc/packages/1.11.6/bin/daemon`, maka daemon sedang berjalan.

   b. Mulai daemon:

   ```
   cd /greengrass/ggc/core/
   sudo ./greengrassd start
   ```

2. Buat deployment.

   - Ganti *`grup-id`* dengan `Id` yang Anda salin untuk grup.
   - Ganti*`group-version-id`*dengan`Version`yang Anda salin untuk versi grup baru.

   ```
   aws greengrass create-deployment \
   --deployment-type NewDeployment \
   --group-id group-id \
   --group-version-id group-version-id
   ```

3. Salin `DeploymentId` dari output.
4. Dapatkan status deployment.

   - Ganti *`grup-id`* dengan `Id` yang Anda salin untuk grup.
   - Ganti *`id-deployment`* dengan `DeploymentId` yang Anda salin untuk deployment.

   ```
   aws greengrass get-deployment-status \
   ```

```
--group-id group-id \
--deployment-id deployment-id
```

Jika statusnya `Success`, deployment berhasil. Untuk langkah-langkah penyelesaian masalah, lihat Memecahkan masalah.

Uji solusi

1. Pada halaman beranda konsol AWS IoT tersebut, pilih Pengujian.
2. Untuk Berlangganan topik, gunakan nilai berikut, dan lalu pilih Langganan. Konektor Notifikasi Twilio menerbitkan informasi status untuk topik ini.

Properti	Nilai
Topik langganan	twilio/message/status
Tampilan muatan MQTT	Tampilkan muatan sebagai string

3. Untuk Publikasikan ke topik, gunakan nilai-nilai berikut, dan kemudian pilih Publikasikan untuk memanggil fungsi.

Properti	Nilai
Topik	temperatur/input
Message	Ganti*nama penerima* dengan nama dan *recipient-phone-number* dengan nomor telepon dari penerima pesan teks. Contoh: `+12345000000` `{ "to_name": " recipient-name ", "to_number": " recipient-phone-number ", "temperature": 31 }`

Properti	Nilai
	Jika Anda menggunakan akun uji coba, Anda harus menambahkan nomor telepon penerima selain Twilio ke daftar nomor telepon terverifikasi. Untuk informasi lebih lanjut, lihat Verifikasi Nomor Telepon Pribadi.

Jika berhasil, penerima menerima pesan teks dan konsol menampilkan `success` status dari output data.

Sekarang, ubah `temperature` dalam pesan input untuk **29** dan terbitkan. Karena ini adalah kurang dari 30, TempMonitor Fungsi tidak memicu pesan Twilio.

Lihat juga

- Integrasikan dengan layanan dan protokol menggunakan konektor
- the section called “AWS-disediakan konektor Greengrass”
- the section called “Memulai dengan konektor (konsol)”
- AWS Secrets Manager perintah dalam AWS CLI Referensi Perintah
- AWS Identity and Access Management perintah (IAM) dalam AWS CLI Refensi Perintah
- AWS Lambda perintah dalam AWS CLI Referensi Perintah
- AWS IoT Greengrass perintah dalam AWS CLI Referensi Perintah

Greengrass Discovery RESTful API

Semua perangkat klien yang berkomunikasi denganAWS IoT Greengrass core harus menjadi anggota dari grup Greengrass. Setiap grup harus memiliki core Greengrass. The Discovery API mengaktifkan perangkat untuk mengambil informasi yang diperlukan untuk terhubung ke core Greengrass yang berada di grup Greengrass yang sama sebagai perangkat klien. Ketika perangkat klien pertama kali online, perangkat dapat tersambungAWS IoT Greengrass ke dan gunakan Discovery API untuk menemukan:

- Grup yang menjadi miliknya. Perangkat klien dapat menjadi anggota hingga 10 grup.
- Alamat IP dan port untuk core Greengrass dalam grup.
- Sertifikat CA grup, yang dapat digunakan untuk mengautentikasi perangkat core Greengrass.

> **Note**
>
> Perangkat klien juga dapat menggunakan SDKAWS IoT Perangkat untuk menemukan informasi konektivitas untuk core Greengrass. Untuk informasi selengkapnya, lihat AWS IoT Perangkat SDK.

Untuk menggunakan API ini, kirim permintaan HTTP ke titik akhir Discovery API. Misalnya:

```
https://greengrass-ats.iot.region.amazonaws.com:port/greengrass/discover/thing/thing-name
```

Untuk dafta wilayahyang didukung Amazon Web Services Regions dan titik akhir untukAWS IoT Greengrass Discovery API, lihat AWS IoT Greengrasstitik akhir dan kuota di Referensi Umum AWS. Ini adalah hanya bidang data API. Titik akhir untuk manajemen grup dan AWS IoT Core operasi berbeda dari titik akhir Discovery API.

Permintaan

Permintaan berisi header HTTP standar dan dikirim ke titik akhir Greengrass Discovery, seperti yang ditunjukkan dalam contoh berikut.

Nomor port tergantung pada apakah inti dikonfigurasi untuk mengirim lalu lintas HTTPS melalui port 8443 atau port 443. Untuk informasi selengkapnya, lihat the section called “Connect pada port 443 atau melalui proksi jaringan”.

Port 8443

```
HTTP GET https://greengrass-ats.iot.region.amazonaws.com:8443/greengrass/discover/
thing/thing-name
```

Port 443

```
HTTP GET https://greengrass-ats.iot.region.amazonaws.com:443/greengrass/discover/
thing/thing-name
```

Klien yang terhubung pada port 443 harus menerapkan Application Layer Protocol Negotiation (ALPN) TLS ekstensi dan lulus `x-amzn-http-ca` sebagai `ProtocolName` di `ProtocolNameList`. Untuk informasi selengkapnya, lihat Protokol di AWS IoT Panduan Pengembang.

> **Note**
>
> Contoh ini menggunakan titik akhir Amazon Trust Services (ATS), yang digunakan dengan sertifikat ATS root CA (disarankan). Titik akhir harus sesuai dengan jenis sertifikat root CA. Untuk informasi selengkapnya, lihat the section called “Titik akhir Anda harus sesuai dengan jenis sertifikat”.

Response

Setelah berhasil, respon mencakup header HTTP standar ditambah kode dan tubuh berikut:

```
HTTP 200
BODY: response document
```

Untuk informasi selengkapnya, lihat Contoh menemukan dokumen respon.

Otorisasi Discovery

Mengambil informasi konektivitas memerlukan kebijakan yang mengizinkan pemanggil untuk melakukan `greengrass:Discover` tindakan. Autentikasi mutual TLS dengan sertifikat klien adalah satu-satunya bentuk autentikasi yang diterima. Berikut ini adalah contoh kebijakan yang mengizinkan pemanggil untuk melakukan tindakan ini:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": "greengrass:Discover",
        "Resource": ["arn:aws:iot:us-west-2:123456789012:thing/MyThingName"]
     }]
}
```

Contoh menemukan dokumen respon

Dokumen berikut menunjukkan respon untuk perangkat klien yang merupakan anggota dari kelompok dengan satu core Greengrass, satu titik akhir, dan satu grup sertifikat CA:

```
{
  "GGGroups": [
    {
      "GGGroupId": "gg-group-01-id",
      "Cores": [
        {
          "thingArn": "core-01-thing-arn",
          "Connectivity": [
            {
              "id": "core-01-connection-id",
              "hostAddress": "core-01-address",
              "portNumber": core-01-port,
              "metadata": "core-01-description"
            }
          ]
        }
      ],
      "CAs": [
        "-----BEGIN CERTIFICATE-----cert-contents-----END CERTIFICATE-----"
      ]
```

```
    }
  ]
}
```

Dokumen berikut menunjukkan respon untuk perangkat klien yang merupakan anggota dari dua kelompok dengan satu core Greengrass, beberapa titik, dan beberapa grup sertifikat CA:

```
{
  "GGGroups": [
    {
      "GGGroupId": "gg-group-01-id",
      "Cores": [
        {
          "thingArn": "core-01-thing-arn",
          "Connectivity": [
            {
              "id": "core-01-connection-id",
              "hostAddress": "core-01-address",
              "portNumber": core-01-port,
              "metadata": "core-01-connection-1-description"
            },
            {
              "id": "core-01-connection-id-2",
              "hostAddress": "core-01-address-2",
              "portNumber": core-01-port-2,
              "metadata": "core-01-connection-2-description"
            }
          ]
        }
      ],
      "CAs": [
        "-----BEGIN CERTIFICATE-----cert-contents-----END CERTIFICATE-----",
        "-----BEGIN CERTIFICATE-----cert-contents-----END CERTIFICATE-----",
        "-----BEGIN CERTIFICATE-----cert-contents-----END CERTIFICATE-----"
      ]
    },
    {
      "GGGroupId": "gg-group-02-id",
      "Cores": [
         {
            "thingArn":"core-02-thing-arn",
            "Connectivity" : [
            {
```

```
              "id": "core-02-connection-id",
              "hostAddress": "core-02-address",
              "portNumber": core-02-port,
              "metadata": "core-02-connection-1-description"
            }
            ],
            "CAs": [
                "-----BEGIN CERTIFICATE-----cert-contents-----END CERTIFICATE-----",
                "-----BEGIN CERTIFICATE-----cert-contents-----END CERTIFICATE-----",
                "-----BEGIN CERTIFICATE-----cert-contents-----END CERTIFICATE-----"
            ]
        }
    ]
    }
}
```

> **Note**
>
> Sebuah grup Greengrass harus mendefinisikan tepat satu core Greengrass. Setiap respons dari AWS IoT Greengrass layanan yang berisi daftar core Greengrass hanya berisi satu core Greengrass.

Jika Anda telah cURL menginstal, Anda dapat menguji permintaan discovery. Misalnya:

```
$ curl --cert 1a23bc4d56.cert.pem --key 1a23bc4d56.private.key https://greengrass-
ats.iot.us-west-2.amazonaws.com:8443/greengrass/discover/thing/MyDevice
{"GGGroups":[{"GGGroupId":"1234a5b6-78cd-901e-2fgh-3i45j6k1789","Cores":
[{"thingArn":"arn:aws:iot:us-west-2:1234567
89012:thing/MyFirstGroup_Core","Connectivity":
[{"Id":"AUTOIP_192.168.1.4_1","HostAddress":"192.168.1.5","PortNumber
":8883,"Metadata":""}]}],"CAs":["-----BEGIN CERTIFICATE-----\ncert-contents\n-----END
 CERTIFICATE-----\n"]}]}
```

Keamanan di AWS IoT Greengrass

Keamanan cloud di AWS merupakan prioritas tertinggi. Sebagai pelanggan AWS, Anda akan mendapatkan manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara AWS dan Anda. Model tanggung jawab bersama menggambarkan ini sebagai keamanan dari cloud dan keamanan di dalam cloud:

- Keamanan dari cloud – AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan layanan AWS di Cloud AWS Cloud. AWS juga menyediakan layanan yang dapat Anda gunakan dengan aman. Auditor pihak ketiga menguji dan memverifikasi secara berkala efektivitas keamanan kami sebagai bagian dari Program Kepatuhan AWS. Untuk mempelajari tentang program kepatuhan yang berlaku diAWS IoT Greengrass, lihat AWSLayanan dalam Cakupan menurut Program KepatuhanAWS .
- Keamanan dalam cloud – Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, termasuk sensitivitas data Anda, persyaratan perusahaan Anda, serta undang-undang dan peraturan yang berlaku.

Saat Anda menggunakan AWS IoT Greengrass, Anda juga bertanggung jawab untuk mengamankan perangkat Anda, koneksi jaringan lokal, dan kunci privat.

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama ketika menggunakan AWS IoT Greengrass. Topik berikut akan menunjukkan kepada Anda cara membuat konfigurasi AWS IoT Greengrass untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga mempelajari cara menggunakan layanan AWS lain yang membantu Anda memantau dan mengamankan sumber daya AWS IoT Greengrass Anda.

Topik

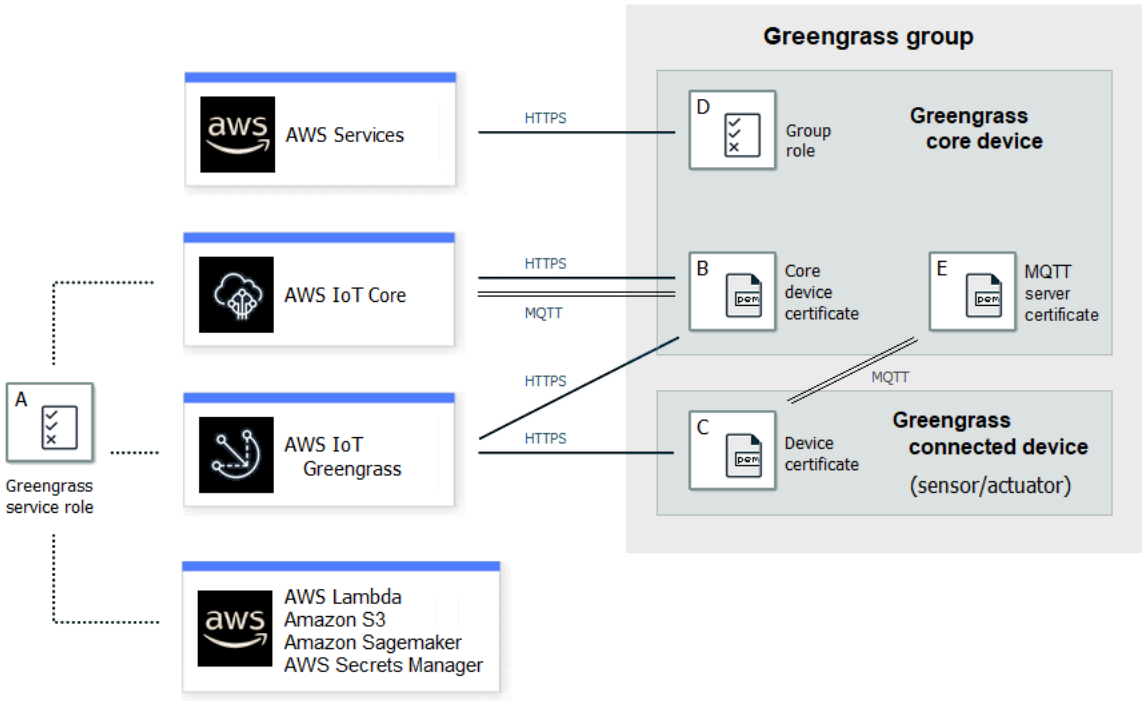
- Ikhtisar AWS IoT Greengrass keamanan
- Perlindungan data di AWS IoT Greengrass
- Autentikasi dan otorisasi perangkat untuk AWS IoT Greengrass
- Identity and access management untuk AWS IoT Greengrass
- Validasi kepatuhan untuk AWS IoT Greengrass
- Ketahanan di AWS IoT Greengrass

- Keamanan infrastruktur dalam AWS IoT Greengrass
- Analisis konfigurasi dan kerentanan dalam AWS IoT Greengrass
- AWS IoT Greengrass dan titik akhir VPC antarmuka (AWS PrivateLink)
- Praktik terbaik keamanan untuk AWS IoT Greengrass

Ikhtisar AWS IoT Greengrass keamanan

AWS IoT Greengrass menggunakan sertifikat, AWS IoT kebijakan, serta kebijakan dan peran IAM X.509 untuk mengamankan aplikasi yang berjalan di perangkat di lingkungan Greengrass lokal Anda.

Diagram berikut menunjukkan komponen model AWS IoT Greengrass keamanan:



A - Peran layanan Greengrass

Peran IAM yang dibuat pelanggan diasumsikan oleh AWS IoT Greengrass saat mengakses AWS sumber daya Anda dari AWS IoT Core, AWS Lambda, dan layanan lainnya. AWS Untuk informasi selengkapnya, lihat the section called “Peran layanan Greengrass”.

B - Sertifikat perangkat core

Sertifikat X.509 yang digunakan untuk mengautentikasi inti Greengrass dengan dan. AWS IoT Core AWS IoT Greengrass Untuk informasi selengkapnya, lihat the section called “Autentikasi dan otorisasi perangkat”.

C - sertifikat perangkat

Sertifikat X.509 yang digunakan untuk mengautentikasi perangkat klien, yang juga dikenal sebagai perangkat yang terhubung, dengan dan. AWS IoT Core AWS IoT Greengrass Untuk informasi selengkapnya, lihat the section called “Autentikasi dan otorisasi perangkat”.

D - Peran grup

Peran IAM yang dibuat pelanggan diasumsikan oleh AWS IoT Greengrass saat memanggil AWS layanan dari inti Greengrass.

Anda menggunakan peran ini untuk menentukan izin akses yang diperlukan oleh fungsi dan konektor Lambda yang ditentukan pengguna untuk AWS mengakses layanan, seperti DynamoDB. Anda juga menggunakannya untuk memungkinkan AWS IoT Greengrass mengekspor aliran manajer aliran ke AWS layanan dan menulis ke CloudWatch Log. Untuk informasi selengkapnya, lihat the section called “Peran grup Greengrass”.

> **Note**
>
> AWS IoT Greengrass tidak menggunakan peran eksekusi Lambda yang ditentukan AWS Lambda untuk versi cloud dari fungsi Lambda.

sertifikat server E - MQTT

Sertifikat yang digunakan untuk otentikasi timbal balik Transport Layer Security (TLS) antara perangkat inti Greengrass dan perangkat klien di grup Greengrass. Sertifikat ditandatangani oleh sertifikat CA grup, yang disimpan dalam AWS Cloud.

Alur kerja koneksi perangkat

Bagian ini menjelaskan bagaimana perangkat klien terhubung ke AWS IoT Greengrass layanan dan perangkat inti Greengrass. Perangkat klien adalah AWS IoT Core perangkat terdaftar yang berada dalam grup Greengrass yang sama dengan perangkat inti.

- Perangkat inti Greengrass menggunakan sertifikat perangkat, kunci pribadi, dan sertifikat CA root AWS IoT Core untuk terhubung ke layanan. AWS IoT Greengrass Pada perangkat core, pada `crypto` objek dalam file konfigurasi menentukan path file untuk item ini.
- Perangkat core Greengrass mengunduh informasi keanggotaan grup dari AWS IoT Greengrass layanan.
- Ketika deployment dibuat untuk perangkat core Greengrass, Device Certificate Manager (DCM) menangani manajemen sertifikat server lokal untuk perangkat core Greengrass.
- Perangkat klien terhubung ke AWS IoT Greengrass layanan menggunakan sertifikat perangkat, kunci pribadi, dan sertifikat AWS IoT Core root CA. Setelah membuat koneksi, perangkat klien menggunakan Layanan Greengrass Discovery untuk menemukan alamat IP perangkat inti Greengrass. Perangkat klien juga mengunduh sertifikat CA grup, yang digunakan untuk otentikasi timbal balik TLS dengan perangkat inti Greengrass.
- Perangkat klien mencoba terhubung ke perangkat inti Greengrass, melewati sertifikat perangkat dan ID kliennya. Jika ID klien cocok dengan nama benda perangkat klien dan sertifikat valid (bagian dari grup Greengrass), koneksi dibuat. Jika tidak, koneksi dihentikan.

AWS IoT Kebijakan untuk perangkat klien harus memberikan `greengrass:Discover` izin untuk mengizinkan perangkat klien menemukan informasi konektivitas untuk inti. Untuk informasi selengkapnya tentang kebijakan laporan, lihat the section called “Otorisasi Discovery”.

Mengkonfigurasi keamanan AWS IoT Greengrass

Untuk mengonfigurasi keamanan aplikasi Greengrass Anda

1. Buat AWS IoT Core sesuatu untuk perangkat inti Greengrass Anda.
2. Menghasilkan pasangan kunci dan sertifikat perangkat untuk perangkat core Greengrass Anda.
3. Buat dan lampirkan AWS IoT kebijakan ke sertifikat perangkat. Sertifikat dan kebijakan memungkinkan akses dan layanan perangkat inti Greengrass. AWS IoT Core AWS IoT Greengrass Untuk informasi selengkapnya, lihat Minimal AWS IoT kebijakan untuk perangkat core.

> **Note**
>
> Penggunaan variabel kebijakan benda (`iot:Connection.Thing.*`) dalam AWS IoT kebijakan untuk perangkat inti tidak didukung. Inti menggunakan sertifikat perangkat

yang sama untuk membuat beberapa koneksi ke AWS IoT Core tetapi ID klien dalam koneksi mungkin tidak sama persis dengan nama hal inti.

4. Buat Peran layanan Greengrass. Peran IAM ini memberi wewenang AWS IoT Greengrass untuk mengakses sumber daya dari AWS layanan lain atas nama Anda. Ini memungkinkan AWS IoT Greengrass untuk melakukan tugas-tugas penting, seperti mengambil AWS Lambda fungsi dan mengelola bayangan perangkat.

   Anda dapat menggunakan peran layanan yang sama di seluruh Wilayah AWS s, tetapi harus dikaitkan dengan peran Anda Akun AWS di setiap Wilayah AWS tempat yang Anda gunakan AWS IoT Greengrass.

5. (Opsional) Buat Peran grup Greengrass. Peran IAM ini memberikan izin ke fungsi dan konektor Lambda yang berjalan pada inti Greengrass untuk memanggil layanan. AWS Misalnya, konektor Firehose Kinesis memerlukan izin untuk menulis catatan ke aliran pengiriman Amazon Data Firehose.

   Anda dapat melampirkan hanya satu peran untuk core Greengrass.

6. Buat AWS IoT Core sesuatu untuk setiap perangkat yang terhubung ke inti Greengrass Anda.

   > **Note**
   >
   > Anda juga dapat menggunakan AWS IoT Core barang dan sertifikat yang ada.

7. Buat sertifikat perangkat, pasangan kunci, dan AWS IoT kebijakan untuk setiap perangkat yang terhubung ke inti Greengrass Anda.

AWS IoT Greengrass prinsip keamanan inti

Inti Greengrass menggunakan prinsip keamanan berikut AWS IoT : klien, server MQTT lokal, dan manajer rahasia lokal. Konfigurasi untuk pelaku utama ini disimpan dalam `crypto` objek dalam `config.json` file konfigurasi. Untuk informasi selengkapnya, lihat the section called “AWS IoT Greengrass file konfigurasi core”.

Konfigurasi ini mencakup jalur ke kunci privat yang digunakan oleh komponen pelaku utama untuk autentikasi dan enkripsi. AWS IoT Greengrass mendukung dua mode penyimpanan kunci privat: berbasis perangkat keras atau berbasis sistem file (default). Untuk informasi lebih lanjut tentang menyimpan kunci pada modul keamanan perangkat keras, lihat the section called “Integrasi keamanan perangkat keras”.

AWS IoT Klien

AWS IoT Klien (klien IoT) mengelola komunikasi melalui internet antara inti Greengrass dan. AWS IoT Core AWS IoT Greengrass menggunakan sertifikat X.509 dengan kunci publik dan pribadi untuk otentikasi timbal balik saat membuat koneksi TLS untuk komunikasi ini. Untuk informasi lebih lanjut, lihat sertifikat X.509 dan AWS IoT Core dalam AWS IoT Core Panduan Developer.

Klien IoT mendukung sertifikat dan kunci RSA dan EC. Sertifikat dan jalur kunci privat ditentukan untuk `IoTCertificate` pelaku utama dalam `config.json`.

Server MQTT

Server MQTT lokal mengelola komunikasi melalui jaringan lokal antara inti Greengrass dan perangkat klien dalam grup. AWS IoT Greengrass menggunakan sertifikat X.509 dengan kunci publik dan pribadi untuk otentikasi timbal balik saat membuat koneksi TLS untuk komunikasi ini.

Secara default, AWS IoT Greengrass menghasilkan kunci pribadi RSA untuk Anda. Untuk mengonfigurasi core untuk menggunakan kunci privat yang berbeda, Anda harus menyediakan path kunci untuk utama `MQTTServerCertificate` di dalam `config.json`. Anda bertanggung jawab untuk memutar kunci yang disediakan pelanggan.

Support kunci privat

	Kunci RSA	Kunci EC
Tipe Kunci	Supported	Supported
Parameter Kunci	Minimum 2048-bit length	NIST P-256 or NIST P-384 curve
Format disk	PKCS#1, PKCS#8	SECG1, PKCS#8
Versi GGC minimum	• Gunakan kunci RSA default: 1.0 • Tentukan kunci RSA: 1.7	• Tentukan kunci EC: 1.9

Konfigurasi kunci privat menentukan proses terkait. Untuk daftar chiper suite yang core Greengrass mendukung sebagai server, lihat the section called “TLS cipher suite mendukung”.

Jika tidak ada kunci privat yang ditentukan (default)

- AWS IoT Greengrass memutar kunci berdasarkan pengaturan rotasi Anda.

- core menghasilkan kunci RSA, yang digunakan untuk menghasilkan sertifikat.
- Sertifikat server MQTT memiliki kunci publik RSA dan tanda tangan SHA-256 RSA.

Jika kunci privat RSA ditentukan (membutuhkan GGC v1.7 atau yang lebih baru)

- Anda bertanggung jawab untuk memutar kunci.
- core menggunakan kunci yang ditentukan untuk menghasilkan sertifikat.
- Kunci RSA harus memiliki panjang minimum 2048 bit.
- Sertifikat server MQTT memiliki kunci publik RSA dan tanda tangan SHA-256 RSA.

Jika kunci privat EC ditentukan (membutuhkan GGC v1.9 atau lebih baru)

- Anda bertanggung jawab untuk memutar kunci.
- core menggunakan kunci yang ditentukan untuk menghasilkan sertifikat.
- Kunci privat EC harus menggunakan kurva NIST P-256 atau NIST P-384.
- Sertifikat server MQTT memiliki kunci publik EC dan tanda tangan SHA-256 RSA.

  Sertifikat server MQTT yang disajikan oleh core memiliki tanda tangan SHA-256 RSA, terlepas dari tipe kunci. Untuk alasan ini, klien harus mendukung validasi sertifikat SHA-256 RSA untuk membuat koneksi aman dengan core.

Secrets Manager

Manajer rahasia lokal dengan aman mengelola salinan rahasia lokal yang Anda buat. AWS Secrets Manager Menggunakan kunci privat untuk mengamankan kunci data yang digunakan untuk mengenkripsi rahasia. Untuk informasi selengkapnya, lihat Men-deploy rahasia ke core.

Secara default, kunci privat klien IoT digunakan, tetapi Anda dapat menentukan kunci privat yang berbeda untuk `SecretsManager` utama pada `config.json`. Hanya tipe kunci RSA yang didukung. Untuk informasi selengkapnya, lihat the section called “Tentukan kunci privat untuk enkripsi rahasia”.

> **Note**
>
> Saat ini, hanya AWS IoT Greengrass mendukung mekanisme padding PKCS #1 v1.5 untuk enkripsi dan dekripsi rahasia lokal saat menggunakan kunci pribadi berbasis perangkat keras. Jika Anda mengikuti instruksi yang disediakan vendor untuk membuat kunci pribadi berbasis perangkat keras secara manual, pastikan untuk memilih PKCS #1 v1.5. AWS IoT Greengrass tidak mendukung Padding Enkripsi Asimetris Optimal (OAEP).

Support kunci privat

	Kunci RSA	Kunci EC
Tipe Kunci	Supported	Not supported
Parameter Kunci	Minimum 2048-bit length	Not applicable
Format disk	PKCS#1, PKCS#8	Not applicable
Versi GGC minimum	1.7	Not applicable

Langganan yang dikelola di dalam alur kerja pesan MQTT

AWS IoT Greengrass menggunakan tabel langganan untuk menentukan bagaimana pesan MQTT dapat dipertukarkan antara perangkat klien, fungsi, dan konektor dalam grup Greengrass, dan dengan atau layanan bayangan lokal. AWS IoT Core Setiap langganan menentukan sumber, target, dan topik MQTT (atau subjek) di mana pesan dikirim atau diterima. AWS IoT Greengrass memungkinkan pesan dikirim dari sumber ke target hanya jika langganan yang sesuai ditentukan.

Langganan menentukan aliran pesan dalam satu arah saja, dari sumber ke target. Untuk mendukung pertukaran pesan dua arah, Anda harus membuat dua langganan, satu untuk setiap arah.

TLS cipher suite mendukung

AWS IoT Greengrass menggunakan model keamanan AWS IoT Core transportasi untuk mengenkripsi komunikasi dengan cloud dengan menggunakan suite cipher TLS. Selain itu, AWS IoT Greengrass data dienkripsi saat istirahat (di cloud). Untuk informasi selengkapnya tentang keamanan AWS IoT Core transportasi dan cipher suite yang didukung, lihat Keamanan transportasi di Panduan AWS IoT Core Pengembang.

Cipher Suite yang Didukung untuk Komunikasi Jaringan Lokal

Berbeda dengan AWS IoT Core, AWS IoT Greengrass inti mendukung rangkaian cipher TLS jaringan lokal berikut untuk algoritma penandatanganan sertifikat. Semua chiper suites ini didukung ketika kunci privat disimpan pada sistem file. Subset didukung ketika core dikonfigurasi untuk menggunakan modul keamanan perangkat keras (HSM). Untuk informasi lebih lanjut, lihat the section called “Keamanan utama” dan the section called “Integrasi keamanan perangkat keras”. Tabel ini juga mencakup versi minimum perangkat lunak AWS IoT Greengrass Core yang diperlukan untuk dukungan.

	Cipher	Support HSM	Versi GGC minimum
TLSv1.2	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Supported	1.0
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	Supported	1.0
	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	Supported	1.0
	TLS_RSA_WITH_AES_128_CBC_SHA	Not supported	1.0
	TLS_RSA_WITH_AES_128_GCM_SHA256	Not supported	1.0
	TLS_RSA_WITH_AES_256_CBC_SHA	Not supported	1.0
	TLS_RSA_WITH_AES_256_GCM_SHA384	Not supported	1.0
	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	Supported	1.9
	TLS_ECDHE_ECDSA_WI	Supported	1.9

	Cipher	Support HSM	Versi GGC minimum
	TH_AES_25 6_GCM_SHA384		
TLSv1.1	TLS_ECDHE _RSA_WITH _AES_128_CBC_SHA	Supported	1.0
	TLS_ECDHE _RSA_WITH _AES_256_CBC_SHA	Supported	1.0
	TLS_RSA_W ITH_AES_1 28_CBC_SHA	Not supported	1.0
	TLS_RSA_W ITH_AES_2 56_CBC_SHA	Not supported	1.0
TLSv1.0	TLS_ECDHE _RSA_WITH _AES_128_CBC_SHA	Supported	1.0
	TLS_ECDHE _RSA_WITH _AES_256_CBC_SHA	Supported	1.0
	TLS_RSA_W ITH_AES_1 28_CBC_SHA	Not supported	1.0
	TLS_RSA_W ITH_AES_2 56_CBC_SHA	Not supported	1.0

Perlindungan data di AWS IoT Greengrass

Model tanggung jawab bersama AWS diterapkan untuk perlindungan data AWS IoT Greengrass. Sebagaimana dijelaskan dalam model ini, AWS bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk memelihara kendali atas isi yang dihost pada infrastruktur ini. Anda juga bertanggung jawab atas tugas-tugas konfigurasi dan manajemen keamanan untuk Layanan AWS yang Anda gunakan. Lihat informasi yang lebih lengkap tentang privasi data dalam Pertanyaan Umum Privasi Data. Lihat informasi tentang perlindungan data di Eropa di pos blog Model Tanggung Jawab Bersama dan GDPR AWS di Blog Keamanan AWS.

Untuk tujuan perlindungan data, sebaiknya lindungi kredensial Akun AWS dan siapkan untuk masing-masing pengguna AWS IAM Identity Center atau AWS Identity and Access Management (IAM). Dengan cara itu, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugasnya. Kami juga menyarankan supaya Anda mengamankan data dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk berkomunikasi dengan sumber daya AWS. Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan pengelogan aktivitas pengguna dengan AWS CloudTrail.
- Gunakan solusi enkripsi AWS, bersama semua kontrol keamanan bawaan dalam Layanan AWS.
- Gunakan layanan keamanan terkelola lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-2 ketika mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi yang lebih lengkap tentang titik akhir FIPS yang tersedia di Standar Pemrosesan Informasi Federal (FIPS) 140-2.

Kami sangat merekomendasikan agar Anda tidak pernah memasukkan informasi identifikasi yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam tanda atau bidang isian bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan AWS IoT Greengrass atau lainnya Layanan AWS menggunakan konsol, APIAWS CLI, atau AWS SDK. Data apa pun yang Anda masukkan ke dalam tanda atau bidang isian bebas yang digunakan untuk nama dapat digunakan untuk log penagihan atau log diagnostik. Saat Anda memberikan URL ke server eksternal, kami sangat menganjurkan supaya Anda tidak menyertakan informasi kredensial di dalam URL untuk memvalidasi permintaan Anda ke server itu.

Untuk informasi selengkapnya tentang perlindungan informasi sensitif di AWS IoT Greengrass, lihat the section called “Jangan log informasi sensitif”.

Untuk informasi selengkapnya tentang perlindungan data, lihat posting blog Model Tanggung Jawab Bersama AWS dan Peraturan Perlindungan Data Umum (GDPR) di Blog Keamanan AWS.

Topik

- Enkripsi data
- Integrasi keamanan perangkat keras

Enkripsi data

AWS IoT Greengrass menggunakan enkripsi untuk melindungi data saat transit (melalui internet atau jaringan lokal) dan saat istirahat (disimpan di AWS Cloud).

Perangkat dalam lingkungan AWS IoT Greengrass sering mengumpulkan data yang dikirim ke layanan AWS untuk diproses lebih lanjut. Untuk informasi selengkapnya tentang enkripsi data di layanan AWS lainnya, lihat dokumentasi keamanan untuk layanan tersebut.

Topik

- Enkripsi dalam transit
- Enkripsi saat tidak aktif
- Manajemen kunci untuk perangkat inti Greengrass

Enkripsi dalam transit

AWS IoT Greengrass memiliki tiga mode komunikasi di mana data berada di dalam transit:

- the section called “Data dalam transit melalui internet”. Komunikasi antara core Greengrass dan AWS IoT Greengrass melalui internet dienkripsi.
- the section called “Data dalam transit melalui jaringan lokal”. Komunikasi antara core Greengrass dan perangkat melalui jaringan lokal dienkripsi.
- the section called “Data pada perangkat core”. Komunikasi antar komponen pada perangkat inti Greengrass tidak dienkripsi.

Data dalam transit melalui internet

AWS IoT Greengrass menggunakan Transport Layer Security (TLS) untuk mengenkripsi semua komunikasi melalui internet. Semua data yang dikirim ke AWS Cloud dikirim melalui koneksi TLS menggunakan protokol MQTT atau HTTPS, sehingga aman secara default. AWS IoT Greengrass menggunakan AWS IoT model keamanan transportasi. Untuk informasi lebih lanjut, lihat Keamanan angkutan di dalam AWS IoT Core Panduan Developer.

Data dalam transit melalui jaringan lokal

AWS IoT Greengrassmenggunakan TLS untuk mengenkripsi semua komunikasi melalui jaringan lokal antara core Greengrass dan perangkat klien. Untuk informasi lebih lanjut, lihat Cipher Suites yang didukung untuk Komunikasi Jaringan Lokal.

Ini adalah tanggung jawab Anda untuk melindungi jaringan lokal dan kunci privat.

Untuk perangkat core Greengrass, adalah tanggung jawab Anda untuk:

- Memastikan agar kernel terus diperbarui dengan patch keamanan terbaru.
- Memastikan perpustakaan sistem terus diperbarui dengan patch keamanan terbaru.
- Lindungi kunci privat. Untuk informasi selengkapnya, lihat the section called “Manajemen kunci”.

Untuk perangkat klien, adalah tanggung jawab Anda untuk:

- Memastikan TLS stack up to date.
- Lindungi kunci privat.

Data pada perangkat core

AWS IoT Greengrass tidak mengenkripsi data yang dipertukarkan secara lokal pada perangkat core Greengrass karena data tidak meninggalkan perangkat. Ini termasuk komunikasi antara fungsi Lambda yang ditetapkan pengguna, konektor, sebuah AWS IoT Greengrass Core SDK, dan komponen sistem, seperti pengelola pengaliran.

Enkripsi saat tidak aktif

AWS IoT Greengrass menyimpan data Anda:

- the section called “Data at rest di AWS Cloud”. Data ini dienkripsi.
- the section called “Data at rest pada inti Greengrass”. Data ini tidak dienkripsi (kecuali salinan lokal rahasia Anda).

Data at rest di AWS Cloud

AWS IoT Greengrass mengenkripsi data pelanggan yang tersimpan di AWS Cloud. Data ini dilindungi menggunakan kunci AWS KMS yang dikelola oleh AWS IoT Greengrass.

Data at rest pada inti Greengrass

AWS IoT Greengrass bergantung pada izin file Unix dan enkripsi disk penuh (jika diaktifkan) untuk melindungi data at rest yang ada di inti. Ini adalah tanggung jawab Anda untuk mengamankan sistem file dan perangkat.

Namun, AWS IoT Greengrass mengenkripsi salinan lokal rahasia yang diambil dari AWS Secrets Manager. Untuk informasi selengkapnya, lihat the section called “Enkripsi rahasia”.

Manajemen kunci untuk perangkat inti Greengrass

Ini adalah tanggung jawab pelanggan untuk menjamin penyimpanan aman kunci kriptografi (publik dan privat) pada perangkat core Greengrass. AWS IoT Greengrass menggunakan kunci publik dan privat untuk skenario berikut:

- Kunci klien IoT digunakan dengan sertifikat IoT untuk mengautentikasi jabat tangan Transport Layer Security (TLS) ketika core Greengrass menghubungkan ke AWS IoT Core. Untuk informasi selengkapnya, lihat the section called “Autentikasi dan otorisasi perangkat”.

  > **Note**
  >
  > Kunci dan sertifikat juga disebut sebagai kunci privat inti dan sertifikat perangkat inti.

- Kunci server MQTT digunakan sertifikat server MQTT untuk mengotentikasi koneksi TLS antara core dan perangkat klien. Untuk informasi selengkapnya, lihat the section called “Autentikasi dan otorisasi perangkat”.
- Secrets manager lokal juga menggunakan kunci klien IoT untuk melindungi kunci data yang digunakan untuk mengenkripsi rahasia lokal, tetapi Anda dapat menyediakan kunci privat Anda sendiri. Untuk informasi selengkapnya, lihat the section called “Enkripsi rahasia”.

Sebuah core Greengrass mendukung penyimpanan kunci privat menggunakan izin sistem file, modul keamanan perangkat keras, atau keduanya. Jika Anda menggunakan kunci privat berbasis sistem file, Anda bertanggung jawab atas penyimpanan aman mereka pada perangkat core.

Pada core Greengrass, lokasi kunci privat Anda ditentukan dalam `crypto` bagian dari `config.json` file. Jika Anda mengonfigurasi core untuk menggunakan kunci yang disediakan pelanggan untuk sertifikat server MQTT, Anda bertanggung jawab untuk memutar kunci tersebut. Untuk informasi selengkapnya, lihat the section called “Keamanan utama”.

Untuk perangkat klien, Anda bertanggung jawab untuk selalu update tumpukan TLS dan melindungi kunci privat. Kunci privat digunakan dengan sertifikat perangkat untuk mengautentikasi koneksi TLS dengan AWS IoT Greengrass layanan.

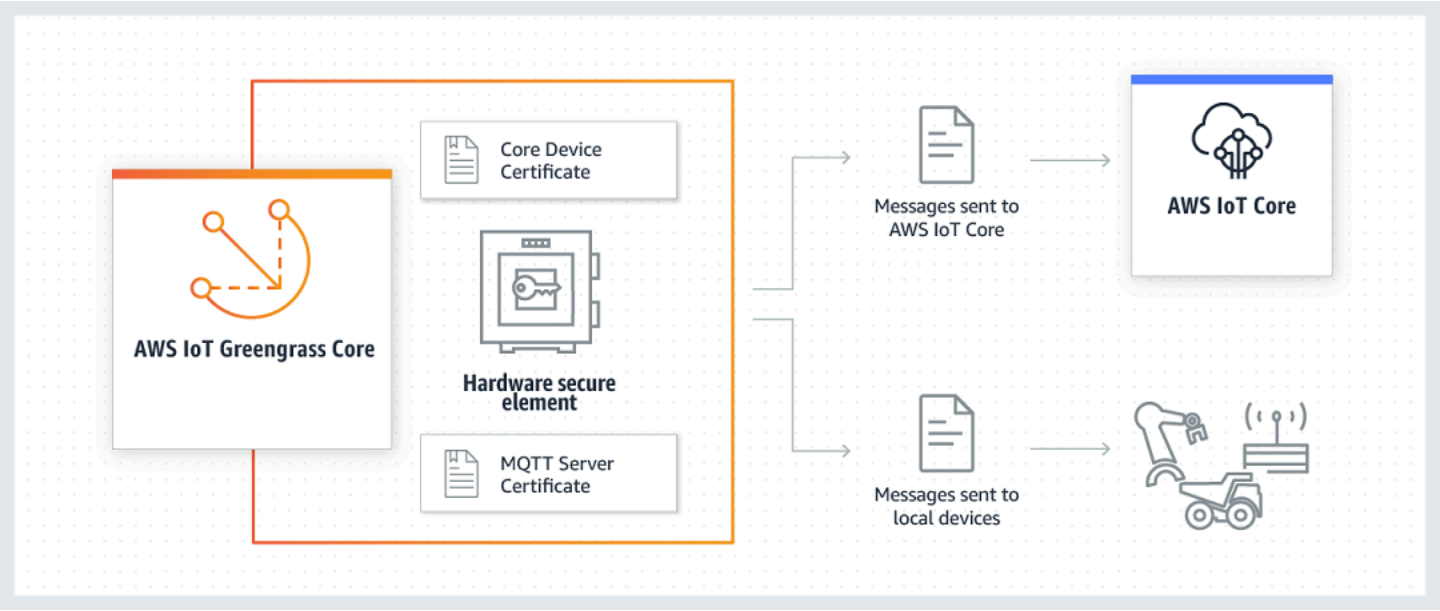
Integrasi keamanan perangkat keras

Fitur ini tersedia untuk AWS IoT Greengrass Core v1.7 dan yang lebih baru.

AWS IoT Greengrass mendukung penggunaan modul keamanan perangkat keras (HSM) melalui antarmuka PKCS #11 untuk penyimpanan aman dan pembongkaran kunci privat. Ini mencegah kunci dari yang terekspos atau terduplikasi dalam perangkat lunak. Kunci privat dapat disimpan dengan aman pada modul perangkat keras, seperti HSMS, Trusted Platform Modules (TPM), atau elemen kriptografi lainnya.

Cari perangkat yang memenuhi syarat untuk fitur ini dalam AWS Partner Katalog Perangkat.

Diagram berikut menunjukkan arsitektur keamanan perangkat keras untuk AWS IoT Greengrass core.



Pada penginstalan standar, AWS IoT Greengrass menggunakan dua kunci privat. Salah satu kunci yang digunakan oleh komponen klien AWS IoT (klien IoT) selama jabat tangan Transport Layer

Security (TLS) ketika core Greengrass menghubungkan ke AWS IoT Core. (Kunci ini juga disebut sebagai kunci privat core.) Kunci lainnya digunakan oleh server MQTT lokal, yang mengizinkan perangkat Greengrass untuk berkomunikasi dengan core Greengrass. Jika Anda ingin menggunakan keamanan perangkat keras untuk kedua komponen, Anda dapat menggunakan kunci privat bersama atau kunci privat terpisah. Untuk informasi selengkapnya, lihat the section called “ Praktik penyediaan”.

> **Note**
>
> Pada penginstalan standar, secrets manager lokal juga menggunakan kunci klien IoT untuk proses enkripsi, tetapi Anda dapat menggunakan kunci privat Anda sendiri. Ini harus menjadi kunci RSA dengan panjang minimum 2048 bit. Untuk informasi selengkapnya, lihat the section called “Tentukan kunci privat untuk enkripsi rahasia”.

Persyaratan

Sebelum Anda dapat mengonfigurasi keamanan perangkat keras untuk core Greengrass, Anda harus memiliki hal berikut:

- Modul keamanan perangkat keras (HSM) yang mendukung konfigurasi kunci privat target Anda untuk klien IoT, server MQTT lokal, dan komponen secrets manager lokal. Konfigurasi dapat mencakup satu, dua, atau tiga kunci privat berbasis perangkat keras, tergantung pada apakah Anda mengonfigurasi komponen untuk berbagi kunci. Untuk informasi lebih lanjut tentang mendukung kunci privat, lihat the section called “Keamanan utama”.
  - Untuk kunci RSA: Ukuran kunci RSA-2048 (atau lebih besar) danPKCS #1 v1.5skema tanda tangan.
  - Untuk kunci EC: Kurva NIST P-256 atau NIST P-384.

  > **Note**
  >
  > Cari perangkat yang memenuhi syarat untuk fitur ini dalam AWS Partner Katalog Perangkat.

- Pustaka penyedia PKCS#11 yang dapat dimuat saat waktu aktif (menggunakan libdl) dan menyediakan fungsi PKCS#11 ini.
- Modul perangkat keras harus dapat diatasi dengan label slot, sebagaimana ditentukan di dalam spesifikasi PKCS#11.

- Kunci privat harus dihasilkan dan dimuat pada HSM dengan menggunakan alat penyediaan yang disediakan vendor.
- Kunci privat harus dapat diatasi dengan label objek.
- Sertifikat perangkat core. Ini adalah sertifikat klien IoT yang sesuai dengan kunci privat.
- Jika Anda menggunakan agen pembaruan Greengrass OTA, pustaka pembungkus OpenSSL libp11 PKCS#11 harus diinstal. Untuk informasi selengkapnya, lihat the section called “Konfigurasi pembaruan OTA”.

Sebagai tambahan, pastikan bahwa kondisi berikut terpenuhi:

- Sertifikat klien IoT yang terkait dengan kunci privat terdaftar di dalam AWS IoT dan diaktifkan. Anda dapat memverifikasi hal ini diAWS IoTkonsol di bawahKelola, perluasSemua perangkat, chooseThingsdan pilihlahSertifikattab untuk hal inti.
- Perangkat lunak Core v1.7 AWS IoT Greengrass atau yang lebih baru diinstal pada perangkat core, seperti yang dideskripsikan di dalam Modul 2 dari tutorial Memulai. Versi 1.9 atau yang lebih baru diperlukan untuk menggunakan kunci EC untuk server MQTT.
- Sertifikat yang melekat pada core Greengrass. Anda dapat memverifikasi ini dari halaman Kelola untuk hal core dalam konsol AWS IoT tersebut.

> **Note**
>
> Saat ini, AWS IoT Greengrass tidak mendukung pemuatan sertifikat CA atau sertifikat klien IoT langsung dari HSM. Sertifikat harus dimuat sebagai file teks biasa pada sistem file di lokasi yang dapat dibaca oleh Greengrass.

Konfigurasi keamanan perangkat keras untuk AWS IoT Greengrass core

Keamanan perangkat keras dikonfigurasi dalam file konfigurasi Greengrass. Ini adalah file `config.json` yang terletak di dalam direktori `/greengrass-root/config` ini.

> **Note**
>
> Untuk menjalani proses pengaturan konfigurasi HSM menggunakan implementasi perangkat lunak murni, lihat the section called “Modul 7: Mensimulasikan integrasi keamanan perangkat keras”.

> **Important**
>
> Konfigurasi simulasi dalam contoh tidak menyediakan manfaat keamanan apa pun. Ini dimaksudkan untuk mengizinkan Anda mempelajari tentang spesifikasi PKCS#11 dan melakukan pengujian awal perangkat lunak Anda jika Anda berencana untuk menggunakan HSM berbasis perangkat keras di masa mendatang.

Untuk mengonfigurasi keamanan perangkat keras di AWS IoT Greengrass, Anda mengedit objek `crypto` di dalam `config.json`.

Saat menggunakan keamanan perangkat keras, objek `crypto` digunakan untuk menentukan path ke sertifikat, kunci privat, dan aset untuk perpustakaan penyedia PKCS#11 pada core, seperti yang ditunjukkan di dalam contoh berikut.

```
"crypto": {
  "PKCS11" : {
    "OpenSSLEngine" : "/path-to-p11-openssl-engine",
    "P11Provider" : "/path-to-pkcs11-provider-so",
    "slotLabel" : "crypto-token-name",
    "slotUserPin" : "crypto-token-user-pin"
  },
  "principals" : {
    "IoTCertificate" : {
      "privateKeyPath" : "pkcs11:object=core-private-key-label;type=private",
      "certificatePath" : "file:///path-to-core-device-certificate"
    },
    "MQTTServerCertificate" : {
      "privateKeyPath" : "pkcs11:object=server-private-key-label;type=private"
    },
    "SecretsManager" : {
      "privateKeyPath": "pkcs11:object=core-private-key-label;type=private"
    }
  },
  "caPath" : "file:///path-to-root-ca"
```

Objek `crypto` berisi properti berikut:

Bidang	Deskripsi	Catatan
`caPath`	Jalur absolut ke CA root AWS IoT ini.	Harus berupa URI file dalam bentuk: `file:///absolute/path/to/file` . **Note** Pastikan bahwa titik akhir Anda sesuai dengan jenis sertifikat Anda.
`PKCS11`		
`OpenSSLEngine`	Tidak wajib. Jalur absolut ke file `.so` mesin OpenSSL untuk mengaktifkan mendukung PKCS#11 pada OpenSSL.	Harus berupa jalur ke file pada sistem file. Properti ini diperlukan jika Anda menggunakan agen pembaruan Greengrass OTA dengan keamanan perangkat keras. Untuk informasi selengkapnya, lihat the section called “Konfigurasi pembaruan OTA”.
`P11Provider`	Jalur absolut ke perpustakaan libdl-loadable implementasi PKCS#11.	Harus berupa jalur ke file pada sistem file.
`slotLabel`	Slot label yang digunakan untuk mengidentifikasi modul perangkat keras.	Harus sesuai dengan spesifikasi label PKCS#11.

Bidang	Deskripsi	Catatan
`slotUserPin`	PIN pengguna yang digunakan untuk mengautentikasi core Greengrass ke modul.	Harus memiliki izin yang memadai untuk melakukan C_Sign dengan kunci privat yang dikonfigurasi.
`principals`		
`IoTCertificate`	The certificate and private key that the core uses to make requests to AWS IoT.	
`IoTCertificate.privateKeyPath`	Jalur ke kunci privat core.	Untuk penyimpanan sistem file, harus berupa URI file dalam bentuk: `file:///`*`absolute/path/to/file`* . Untuk penyimpanan HSM, harus jalur RFC 7512 PKCS#11 yang menentukan label objek.
`IoTCertificate.certificatePath`	Jalur absolut untuk sertifikat perangkat core.	Harus berupa URI file dalam bentuk: `file:///`*`absolute/path/to/file`* .
`MQTTServerCertificate`	Tidak wajib. Kunci privat yang menggunakan core dalam kombinasi dengan sertifikat untuk bertindak sebagai server MQTT atau gateway.	

Bidang	Deskripsi	Catatan
`MQTTServerCertificate .privateKeyPath`	Jalur ke kunci privat server MQTT lokal.	Gunakan nilai ini untuk menentukan kunci privat Anda sendiri untuk server MQTT lokal. Untuk penyimpanan sistem file, harus berupa URI file dalam bentuk: `file:///absolute/path/to/file` . Untuk penyimpanan HSM, harus jalur RFC 7512 PKCS#11 yang menentukan label objek. Jika properti ini dihilangkan, AWS IoT Greengrass memutar kunci berdasarkan pengaturan rotasi Anda. Jika ditentukan, pelanggan bertanggung jawab untuk memutar kunci.
`SecretsManager`	The private key that secures the data key used for encryption. For more information, see Men-deploy rahasia ke core.	

Bidang	Deskripsi	Catatan
`SecretsManager .privateKeyPath`	Jalur ke kunci privat secrets manager lokal.	Hanya kunci RSA yang didukung. Untuk penyimpanan sistem file, harus berupa URI file dalam bentuk: `file:///absolute/path/to/file` . Untuk penyimpanan HSM, harus jalur RFC 7512 PKCS#11 yang menentukan label objek. Kunci privat harus dibuat menggunakan mekanisme padding PKCS#1 v1.5 ini.

Bidang	Deskripsi	Catatan
`caPath`	Jalur absolut ke CA root AWS IoT ini.	Harus berupa URI file dalam bentuk: `file:///absolute/path/to/file` . **Note** Pastikan bahwa titik akhir Anda sesuai dengan jenis sertifikat Anda.
`PKCS11`		

Bidang	Deskripsi	Catatan
`OpenSSLEngine`	Tidak wajib. Jalur absolut ke file `.so` mesin OpenSSL untuk mengaktifkan mendukung PKCS#11 pada OpenSSL.	Harus berupa jalur ke file pada sistem file. Properti ini diperlukan jika Anda menggunakan agen pembaruan Greengrass OTA dengan keamanan perangkat keras. Untuk informasi selengkapnya, lihat the section called “Konfigurasi pembaruan OTA”.
`P11Provider`	Jalur absolut ke perpustakaan libdl-loadable implementasi PKCS#11.	Harus berupa jalur ke file pada sistem file.
`slotLabel`	Slot label yang digunakan untuk mengidentifikasi modul perangkat keras.	Harus sesuai dengan spesifikasi label PKCS#11.
`slotUserPin`	PIN pengguna yang digunakan untuk mengautentikasi core Greengrass ke modul.	Harus memiliki izin yang memadai untuk melakukan C_Sign dengan kunci privat yang dikonfigurasi.
`principals`		
`IoTCertificate`	The certificate and private key that the core uses to make requests to AWS IoT.	

Bidang	Deskripsi	Catatan
`IoTCertificate.privateKeyPath`	Jalur ke kunci privat core.	Untuk penyimpanan sistem file, harus berupa URI file dalam bentuk: `file:///absolute/path/to/file` . Untuk penyimpanan HSM, harus jalur RFC 7512 PKCS#11 yang menentukan label objek.
`IoTCertificate.certificatePath`	Jalur absolut untuk sertifikat perangkat core.	Harus berupa URI file dalam bentuk: `file:///absolute/path/to/file` .
`MQTTServerCertificate`	Tidak wajib. Kunci privat yang menggunakan core dalam kombinasi dengan sertifikat untuk bertindak sebagai server MQTT atau gateway.	

Bidang	Deskripsi	Catatan
`MQTTServerCertificate .privateKeyPath`	Jalur ke kunci privat server MQTT lokal.	Gunakan nilai ini untuk menentukan kunci privat Anda sendiri untuk server MQTT lokal. Untuk penyimpanan sistem file, harus berupa URI file dalam bentuk: `file:///absolute/path/to/file` . Untuk penyimpanan HSM, harus jalur RFC 7512 PKCS#11 yang menentukan label objek. Jika properti ini dihilangkan, AWS IoT Greengrass memutar kunci berdasarkan pengaturan rotasi Anda. Jika ditentukan, pelanggan bertanggung jawab untuk memutar kunci.
`SecretsManager`	The private key that secures the data key used for encryption. For more information, see Men-deploy rahasia ke core.	

Bidang	Deskripsi	Catatan
`SecretsManager .privateKeyPath`	Jalur ke kunci privat secrets manager lokal.	Hanya kunci RSA yang didukung. Untuk penyimpanan sistem file, harus berupa URI file dalam bentuk: `file:///absolute/path/to/file` . Untuk penyimpanan HSM, harus jalur RFC 7512 PKCS#11 yang menentukan label objek. Kunci privat harus dibuat menggunakan mekanisme padding PKCS#1 v1.5 ini.

Bidang	Deskripsi	Catatan
`caPath`	Jalur absolut ke CA root AWS IoT ini.	Harus berupa URI file dalam bentuk: `file:///absolute/path/to/file` . **Note** Pastikan bahwa titik akhir Anda sesuai dengan jenis sertifikat Anda.
`PKCS11`		

Bidang	Deskripsi	Catatan
`OpenSSLEngine`	Tidak wajib. Jalur absolut ke file `.so` mesin OpenSSL untuk mengaktifkan mendukung PKCS#11 pada OpenSSL.	Harus berupa jalur ke file pada sistem file. Properti ini diperlukan jika Anda menggunakan agen pembaruan Greengrass OTA dengan keamanan perangkat keras. Untuk informasi selengkapnya, lihat the section called “Konfigurasi pembaruan OTA”.
`P11Provider`	Jalur absolut ke perpustakaan libdl-loadable implementasi PKCS#11.	Harus berupa jalur ke file pada sistem file.
`slotLabel`	Slot label yang digunakan untuk mengidentifikasi modul perangkat keras.	Harus sesuai dengan spesifikasi label PKCS#11.
`slotUserPin`	PIN pengguna yang digunakan untuk mengautentikasi core Greengrass ke modul.	Harus memiliki izin yang memadai untuk melakukan C_Sign dengan kunci privat yang dikonfigurasi.
`principals`		
`IoTCertificate`	The certificate and private key that the core uses to make requests to AWS IoT.	

Bidang	Deskripsi	Catatan
`IoTCertificate.privateKeyPath`	Jalur ke kunci privat core.	Untuk penyimpanan sistem file, harus berupa URI file dalam bentuk: `file:///absolute/path/to/file` . Untuk penyimpanan HSM, harus jalur RFC 7512 PKCS#11 yang menentukan label objek.
`IoTCertificate.certificatePath`	Jalur absolut untuk sertifikat perangkat core.	Harus berupa URI file dalam bentuk: `file:///absolute/path/to/file` .
`MQTTServerCertificate`	Tidak wajib. Kunci privat yang menggunakan core dalam kombinasi dengan sertifikat untuk bertindak sebagai server MQTT atau gateway.	

Bidang	Deskripsi	Catatan
`MQTTServerCertificate .privateKeyPath`	Jalur ke kunci privat server MQTT lokal.	Gunakan nilai ini untuk menentukan kunci privat Anda sendiri untuk server MQTT lokal. Untuk penyimpanan sistem file, harus berupa URI file dalam bentuk: `file:///absolute/path/to/file` . Untuk penyimpanan HSM, harus jalur RFC 7512 PKCS#11 yang menentukan label objek. Jika properti ini dihilangkan, AWS IoT Greengrass memutar kunci berdasarkan pengaturan rotasi Anda. Jika ditentukan, pelanggan bertanggung jawab untuk memutar kunci.
`SecretsManager`	The private key that secures the data key used for encryption. For more information, see Men-deploy rahasia ke core.	

Bidang	Deskripsi	Catatan
`SecretsManager .privateKeyPath`	Jalur ke kunci privat secrets manager lokal.	Hanya kunci RSA yang didukung. Untuk penyimpanan sistem file, harus berupa URI file dalam bentuk: `file:///absolute/path/to/file` . Untuk penyimpanan HSM, harus jalur RFC 7512 PKCS#11 yang menentukan label objek. Kunci privat harus dibuat menggunakan mekanisme padding PKCS#1 v1.5 ini.

Praktik penyediaan untuk AWS IoT Greengrass keamanan perangkat keras

Berikut ini adalah praktik penyediaan terkait keamanan dan performa.

Keamanan

- Hasilkan kunci privat langsung pada HSM dengan menggunakan generator nomor acak perangkat keras internal.

  > **Note**
  >
  > Jika Anda mengonfigurasi kunci privat untuk digunakan dengan fitur ini (dengan mengikuti petunjuk yang disediakan oleh vendor perangkat keras), perhatikan bahwa AWS IoT Greengrass saat ini hanya mendukung mekanisme padding PKCS1 v1.5 untuk enkripsi dan dekripsi local secrets. AWS IoT Greengrass tidak mendukung Optimal Asymmetric Encryption Padding (OAEP).

- Konfigurasi kunci privat untuk melarang ekspor.

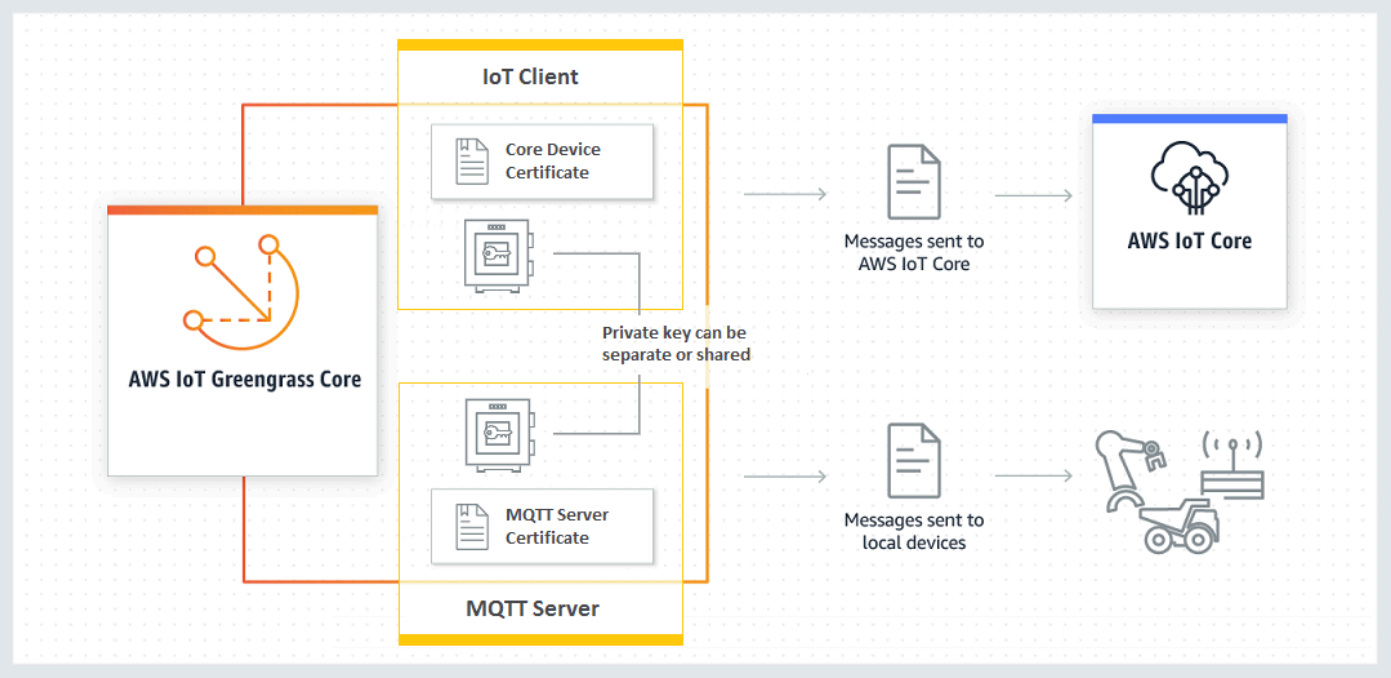
- Gunakan alat penyediaan yang disediakan oleh vendor perangkat keras untuk menghasilkan sertifikat permintaan penandatanganan (CSR) menggunakan kunci privat yang dilindungi perangkat keras, lalu gunakan konsol AWS IoT untuk menghasilkan sertifikat klien.

> **Note**
>
> Praktik tombol berputar tidak berlaku ketika kunci privat dihasilkan pada HSM.

Kinerja

Diagram berikut menunjukkan komponen klien IoT dan server MQTT lokal pada AWS IoT Greengrass core. Jika Anda ingin menggunakan konfigurasi HSM untuk kedua komponen, Anda dapat menggunakan kunci privat yang sama atau kunci privat terpisah. Jika Anda menggunakan kunci terpisah, kunci tersebut harus disimpan di dalam slot yang sama.

> **Note**
>
> AWS IoT Greengrass tidak memaksakan batasan pada jumlah kunci yang Anda simpan di HSM, sehingga Anda dapat menyimpan kunci privat untuk klien IoT, server MQTT, dan komponen secrets manager. Namun, beberapa vendor HSM mungkin memberlakukan batas pada jumlah kunci yang dapat Anda simpan di dalam slot.



Secara umum, kunci klien IoT tidak digunakan sangat sering karena perangkat lunak core AWS IoT Greengrass mempertahankan koneksi berumur panjang ke cloud. Namun, kunci server MQTT digunakan setiap kali perangkat Greengrass terhubung ke core. Interaksi ini secara langsung memengaruhi performa.

Ketika kunci server MQTT disimpan pada HSM, tingkat di mana perangkat dapat terhubung tergantung pada jumlah operasi tanda tangan RSA per detik yang HSM dapat lakukan. Sebagai contoh, jika HSM membutuhkan waktu 300 milidetik untuk melakukan tanda tangan RSASSA-PKCS1-v1.5 pada kunci privat RSA-2048, maka hanya tiga perangkat yang dapat terhubung ke core Greengrass per detik. Setelah koneksi dibuat, HSM tidak lagi digunakan dan standar kuota untuk AWS IoT Greengrass berlaku.

Untuk memitigasi performa bottleneck, Anda dapat menyimpan kunci privat untuk server MQTT pada sistem file bukan pada HSM. Dengan konfigurasi ini, server MQTT berperilaku seolah-olah keamanan perangkat keras tidak diaktifkan.

AWS IoT Greengrass mendukung beberapa konfigurasi penyimpanan-kunci untuk klien IoT dan komponen server MQTT, sehingga Anda dapat mengoptimalkan untuk persyaratan keamanan dan performa Anda. Tabel berikut mencakup konfigurasi contoh.

Konfigurasi	Kunci IoT	Kunci MQTT	Performa
Kunci bersama HSM	HSM: Kunci A	HSM: Kunci A	Dibatasi oleh HSM atau CPU
Kunci terpisah HSM	HSM: Kunci A	HSM: Kunci B	Dibatasi oleh HSM atau CPU
HSM hanya untuk IoT	HSM: Kunci A	Sistem file: Kunci B	Dibatasi oleh CPU
Warisan	Sistem file: Kunci A	Sistem file: Kunci B	Dibatasi oleh CPU

Untuk mengonfigurasi core Greengrass menggunakan kunci berbasis sistem file untuk server MQTT, hilangkan bagian `principals.MQTTServerCertificate` dari `config.json` (atau tentukan path berbasis file ke kunci jika anda tidak menggunakan kunci default yang dihasilkan oleh AWS IoT Greengrass). Objek `crypto` yang dihasilkan terlihat seperti ini:

```
"crypto": {
  "PKCS11": {
    "OpenSSLEngine": "...",
    "P11Provider": "...",
    "slotLabel": "...",
    "slotUserPin": "..."
  },
  "principals": {
    "IoTCertificate": {
      "privateKeyPath": "...",
      "certificatePath": "..."
    },
    "SecretsManager": {
      "privateKeyPath": "..."
    }
  },
  "caPath" : "..."
}
```

Chiper suites yang didukung untuk integrasi keamanan perangkat keras

AWS IoT Greengrass mendukung satu set cipher suite ketika core dikonfigurasi untuk keamanan perangkat keras. Ini adalah bagian dari cipher suites yang didukung ketika core dikonfigurasi untuk menggunakan keamanan berbasis file. Untuk informasi selengkapnya, lihat the section called “TLS cipher suite mendukung”.

> **Note**
>
> Ketika menghubungkan ke core Greengrass dari perangkat Greengrass melalui jaringan lokal, pastikan untuk menggunakan salah satu cipher suites yang didukung untuk membuat koneksi TLS.

Konfigurasikan dukungan untuk over-the-air pembaruan

Untuk mengaktifkan over-the-air (OTA) pembaruanAWS IoT GreengrassPerangkat lunak inti saat menggunakan keamanan perangkat keras, Anda harus menginstal OpenSC libp11Pustaka pembungkus PKCS #11dan edit file konfigurasi Greengrass. Untuk informasi lebih lanjut tentang pembaruan OS, lihat Perbarui OTA AWS IoT Greengrass Perangkat lunak Core.

1. Hentikan daemon Greengrass.

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

   > **Note**
   >
   > *greengrass-root* mewakili jalur di mana AWS IoT Greengrass perangkat lunak Core diinstal pada perangkat Anda. Biasanya, ini adalah `/greengrass` direktori.

2. Pasang mesin OpenSSL. OpenSSL 1.0 atau 1.1 didukung.

   ```
   sudo apt-get install libengine-pkcs11-openssl
   ```

3. Temukan path ke mesin OpenSSL (`libpkcs11.so`) pada sistem Anda:

   a. Dapatkan daftar paket yang diinstal untuk perpustakaan.

```
sudo dpkg -L libengine-pkcs11-openssl
```

File `libpkcs11.so` terletak di `engines` direktori.

b. Salin jalur lengkap ke file (sebagai contoh, `/usr/lib/ssl/engines/libpkcs11.so`).

4. Buka file konfigurasi Greengrass. Ini adalah file `config.json` dalam direktori */greengrass-root*`/config` ini.
5. Untuk properti `OpenSSLEngine` ini, masukkan path ke `libpkcs11.so` file.

```
{
 "crypto": {
   "caPath" : "file:///path-to-root-ca",
   "PKCS11" : {
     "OpenSSLEngine" : "/path-to-p11-openssl-engine",
     "P11Provider" : "/path-to-pkcs11-provider-so",
     "slotLabel" : "crypto-token-name",
     "slotUserPin" : "crypto-token-user-pin"
    },
    ...
  }
  ...
}
```

> **Note**
>
> Jika properti `OpenSSLEngine` tidak ada di dalam objek `PKCS11` ini, maka tambahkan.

6. Mulai daemon Greengrass.

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

Kompatibilitas mundur dengan versi sebelumnya AWS IoT Greengrass perangkat lunak core

Perangkat lunak core AWS IoT Greengrass dengan mendukung keamanan perangkat keras sepenuhnya kompatibel mundur dengan file `config.json` yang dihasilkan untuk v1.6 dan sebelumnya. Jika objek `crypto` tidak ada di dalam file konfigurasi `config.json` ini, maka AWS

IoT Greengrass menggunakan properti `coreThing.certPath`, `coreThing.keyPath`, dan `coreThing.caPath` berbasis file. Kompatibilitas mundur ini berlaku untuk pembaruan Greengrass OTA, yang tidak menimpa konfigurasi berbasis file yang ditentukan di dalam `config.json`.

Perangkat keras tanpa mendukung PKCS#11

Perpustakaan PKCS #11 biasanya disediakan oleh vendor perangkat keras atau open source. Sebagai contoh, dengan perangkat keras yang sesuai standar (seperti TPM1.2), dimungkinkan untuk menggunakan perangkat lunak sumber terbuka yang ada. Namun, jika perangkat keras Anda tidak memiliki implementasi perpustakaan PKCS#11 yang sesuai, atau jika Anda ingin menulis penyedia PKCS#11 kustom, Anda harus kontak perwakilan Support Korporasi AWS dengan pertanyaan terkait integrasi.

Lihat juga

- Panduan Penggunaan Antarmuka Token Kriptografi PKCS #11 Versi 2.40. Diedit oleh John Leiseboer dan Robert Griffin. 16 November 2014. Catatan Komite OASIS 02. http://docs.oasis-open.org/pkcs11/pkcs11-ug/v2.40/cn02/pkcs11-ug-v2.40-cn02.html. Versi terbaru: http://docs.oasis-open.org/pkcs11/pkcs11-ug/v2.40/pkcs11-ug-v2.40.html.
- RFC 7512
- PKCS #1: Enkripsi RSA Versi 1.5

Autentikasi dan otorisasi perangkat untuk AWS IoT Greengrass

Perangkat di lingkungan AWS IoT Greengrass menggunakan sertifikat X.509 untuk autentikasi dan kebijakan AWS IoT untuk otorisasi. Sertifikat dan kebijakan memungkinkan perangkat dengan aman terhubung satu sama lain, AWS IoT Core, dan AWS IoT Greengrass.

Sertifikat X.509 adalah sertifikat digital yang menggunakan standar infrastruktur kunci publik X.509 untuk mengaitkan kunci publik dengan identitas yang terdapat dalam sertifikat. Sertifikat X.509 dikeluarkan oleh entitas terpercaya yang disebut otoritas sertifikasi (CA). CA mempertahankan satu atau lebih sertifikat khusus yang disebut sertifikat CA yang digunakannya untuk mengeluarkan sertifikat X.509. Hanya otoritas sertifikat yang memiliki akses ke sertifikat CA.

Kebijakan AWS IoT menentukan serangkaian operasi yang diperbolehkan untuk perangkat AWS IoT. Secara khusus, kebijakan ini mengizinkan dan menolak akses ke operasi bidang data AWS IoT Core dan AWS IoT Greengrass, seperti menerbitkan pesan MQTT dan mengambil bayangan perangkat.

Semua peranti memerlukan entri di AWS IoT Core registri dan sertifikat X.509 yang diaktifkan dengan kebijakan AWS IoT yang dilampirkan. Perangkat dibagi menjadi dua kategori:

- core Greengrass. Perangkat core Greengrass menggunakan sertifikat dan kebijakan AWS IoT untuk connect ke AWS IoT Core. Sertifikat dan kebijakan juga mengizinkan AWS IoT Greengrass untuk men-deploy informasi konfigurasi, fungsi Lambda, konektor, dan langganan terkelola ke perangkat core.
- Perangkat klien. Perangkat klien (juga disebut perangkat yang terhubung, perangkat Greengrass, atau perangkat) adalah perangkat yang terhubung ke inti Greengrass melalui MQTT. Mereka menggunakan sertifikat dan kebijakan untuk terhubung ke AWS IoT Core dan AWS IoT Greengrass layanan. Hal ini memungkinkan perangkat klien untuk menggunakan AWS IoT Greengrass Discovery Service untuk menemukan dan terhubung ke perangkat inti. Perangkat klien menggunakan sertifikat yang sama untuk terhubung ke gateway AWS IoT Core perangkat dan perangkat inti. Perangkat klien juga menggunakan informasi penemuan untuk autentikasi bersama dengan perangkat inti. Untuk informasi lebih lanjut, lihat the section called “Alur kerja koneksi perangkat” dan the section called “Mengelola autentikasi perangkat dengan core Greengrass”.

Sertifikat X.509

Komunikasi antara perangkat inti dan klien dan antara perangkat dan AWS IoT Core atau AWS IoT Greengrass harus diautentikasi. Autentikasi bersama ini didasarkan pada sertifikat perangkat X.509 terdaftar dan kunci kriptografi.

Dalam AWS IoT Greengrass lingkungan, perangkat menggunakan sertifikat dengan kunci publik dan privat untuk koneksi Keamanan Lapisan Pengangkutan (TLS) berikut:

- Komponen klien AWS IoT pada core Greengrass menghubungkan ke AWS IoT Core dan AWS IoT Greengrass melalui internet.
- Perangkat klien terhubung AWS IoT Greengrass untuk mendapatkan informasi penemuan inti melalui internet.
- Komponen server MQTT pada inti Greengrass yang terhubung ke perangkat klien dalam grup melalui jaringan lokal.

Perangkat lunak Core AWS IoT Greengrass menyimpan sertifikat di dua lokasi:

- Sertifikat perangkat Core di `/greengrass-root/certs`. Biasanya, sertifikat perangkat Core bernama `hash.cert.pem` (sebagai contoh, `86c84488a5.cert.pem`). Sertifikat ini digunakan

oleh klien AWS IoT untuk saling mengautentikasi ketika core menghubungkan ke AWS IoT Core dan layanan AWS IoT Greengrass ini.

- Sertifikat server MTT di */greengrass-root*/ggc/var/state/server. Sertifikat server MQTT bernama server.crt. Sertifikat ini digunakan untuk saling mengautentikasi antara server MQTT lokal (pada Core Greengrass) dan perangkat Greengrass.

> **Note**
>
> *Greengrass-root* mewakili jalur di mana perangkat lunak Core AWS IoT Greengrass diinstal pada perangkat Anda. Biasanya, ini adalah /greengrass direktori.

Untuk informasi selengkapnya, lihat the section called “Keamanan utama”.

Sertifikat Certificate authority (CA)

Perangkat inti dan perangkat klien mengunduh sertifikat CA root yang digunakan untuk otentikasi dengan AWS IoT Core dan AWS IoT Greengrass layanan. Kami merekomendasikan Anda menggunakan sertifikat CA akar Amazon Trust Services (ATS), seperti Amazon Root CA 1. Untuk informasi selengkapnya, lihat Sertifikat CA untuk autentikasi server di Panduan Developer AWS IoT Core.

> **Note**
>
> Jenis sertifikat CA akar Anda harus sesuai dengan titik akhir Anda. Gunakan sertifikat CA root ATS dengan titik akhir ATS (lebih disukai) atau sertifikat VeriSign root CA dengan titik akhir lama. Hanya beberapa dukungan titik akhir warisan Wilayah Amazon Web Services. Untuk informasi selengkapnya, lihat the section called “Titik akhir Anda harus sesuai dengan jenis sertifikat”.

Perangkat klien juga mengunduh sertifikat CA grup Greengrass. Ini digunakan untuk memvalidasi sertifikat server MQTT pada core Greengrass selama autentikasi bersama. Untuk informasi selengkapnya, lihat the section called “Alur kerja koneksi perangkat”. Kedaluwarsa default sertifikat server MQTT adalah tujuh hari.

Rotasi sertifikat pada server MQTT lokal

Perangkat klien menggunakan sertifikat server MQTT lokal untuk otentikasi timbal balik dengan perangkat inti Greengrass. Secara default, sertifikat ini kedaluwarsa dalam tujuh hari. Periode terbatas ini didasarkan pada praktik terbaik keamanan. Sertifikat server MQTT ditandatangani oleh sertifikat CA grup, yang disimpan di cloud.

Agar rotasi sertifikat terjadi, perangkat inti Greengrass Anda harus online dan dapat mengakses layanan secara langsung secara AWS IoT Greengrass teratur. Ketika sertifikat kedaluwarsa, perangkat inti mencoba untuk terhubung ke AWS IoT Greengrass layanan untuk mendapatkan sertifikat baru. Jika koneksi berhasil, perangkat core mengunduh sertifikat server MQTT baru dan memulai ulang layanan MQTT lokal. Pada titik ini, semua perangkat klien yang terhubung ke inti terputus. Jika perangkat inti offline pada saat kedaluwarsa, itu tidak menerima sertifikat pengganti. Upaya baru untuk connect ke perangkat core ditolak. Koneksi yang ada tidak terpengaruh. Perangkat klien tidak dapat terhubung ke perangkat inti sampai koneksi ke AWS IoT Greengrass layanan dipulihkan dan sertifikat server MQTT baru dapat diunduh.

Anda dapat mengatur kedaluwarsa untuk setiap nilai antara 7 dan 30 hari, tergantung pada kebutuhan Anda. Rotasi yang lebih sering memerlukan koneksi cloud yang lebih sering. Rotasi yang kurang sering dapat menimbulkan masalah keamanan. Jika Anda ingin mengatur sertifikat kedaluwarsa ke nilai yang lebih tinggi dari 30 hari, kontak AWS Support.

Dalam konsol AWS IoT tersebut, Anda dapat mengelola sertifikat pada halaman grup pengaturan ini. Di AWS IoT Greengrass API, Anda dapat menggunakan UpdateGroupCertificateConfigurationtindakan.

Ketika sertifikat server MQTT berakhir, setiap upaya untuk memvalidasi sertifikat gagal. Perangkat klien harus dapat mendeteksi kegagalan dan mengakhiri koneksi.

Kebijakan AWS IoT untuk operasi bidang data

Gunakan AWS IoT kebijakan untuk mengesahkan akses ke AWS IoT Core dan AWS IoT Greengrass bidang data. Bidang data AWS IoT Core terdiri dari operasi untuk perangkat, pengguna, dan aplikasi, seperti menghubungkan ke AWS IoT Core dan melanggan topik. Bidang data AWS IoT Greengrass terdiri dari operasi untuk perangkat Greengrass, seperti mengambil deployment dan update informasi konektivitas.

Sebuah kebijakan AWS IoT adalah sebuah dokumen JSON yang mirip dengan Kebijakan IAM. Ini berisi satu atau lebih pernyataan kebijakan yang menentukan properti berikut:

- `Effect`. Mode akses, yang bisa jadi `Allow` atau `Deny`.
- `Action`. Daftar tindakan yang diperbolehkan atau ditolak oleh kebijakan tersebut.
- `Resource`. Daftar sumber daya tempat tindakan tersebut diizinkan atau ditolak.

AWS IoT dukungan kebijakan * sebagai karakter wildcard, dan memperlakukan karakter wildcard MQTT (+ dan #) sebagai string literal. Untuk informasi selengkapnya tentang wildcard *, lihat Menggunakan wildcard di ARN sumber daya di Panduan Pengguna AWS Identity and Access Management.

Untuk informasi lebih lanjut, lihat AWS IoT kebijakan dan AWS IoT tindakan kebijakan di AWS IoT Core Panduan Developer.

> **Note**
>
> AWS IoT Core mengizinkan Anda untuk melampirkan AWS IoT kebijakan untuk grup hal untuk menentukan izin untuk grup perangkat. Kebijakan grup objek tidak mengizinkan akses ke operasi bidang data AWS IoT Greengrass. Untuk mengizinkan suatu objek untuk mengakses operasi bidang data AWS IoT Greengrass, tambahkan izin pada kebijakan AWS IoT yang Anda lampirkan ke sertifikat objek tersebut.

AWS IoT Greengrass tindakan kebijakan

Tindakan Greengrass Core

AWS IoT Greengrass mendefinisikan tindakan kebijakan berikut bahwa perangkat core Greengrass dapat digunakan dalam AWS IoT kebijakan:

`greengrass:AssumeRoleForGroup`

Izin untuk perangkat core Greengrass untuk mengambil kredensial menggunakan fungsi sistem Token Exchange Service (TES) Lambda. Izin yang terkait dengan kredensial diambil didasarkan pada kebijakan yang dilampirkan ke peran grup yang dikonfigurasi.

Izin ini diperiksa ketika perangkat core Greengrass mencoba untuk mengambil kredensial (dengan asumsi kredensial tidak di-cache secara lokal).

`greengrass:CreateCertificate`

Izin untuk perangkat core Greengrass untuk membuat sertifikat server sendiri.

Izin ini diperiksa ketika perangkat core Greengrass menciptakan sertifikat. Perangkat core Greengrass mencoba untuk membuat sertifikat server pada pertama dijalankan, ketika informasi konektivitas perubahan core, dan pada periode rotasi yang ditunjuk.

`greengrass:GetConnectivityInfo`

Izin untuk perangkat core Greengrass untuk mengambil informasi konektivitas sendiri.

Izin ini diperiksa ketika perangkat core Greengrass mencoba untuk mengambil informasi konektivitas dari AWS IoT Core.

`greengrass:GetDeployment`

Izin untuk perangkat core Greengrass untuk mengambil deployment.

Izin ini diperiksa ketika perangkat core Greengrass mencoba untuk mengambil deployment dan deployment status dari cloud.

`greengrass:GetDeploymentArtifacts`

Izin untuk perangkat core Greengrass untuk mengambil artefak deployment seperti informasi grup atau fungsi Lambda.

Izin ini diperiksa ketika perangkat Greengrass grup menerima deployment lalu mencoba untuk mengambil artefak deployment.

`greengrass:UpdateConnectivityInfo`

Izin untuk perangkat core Greengrass untuk update informasi konektivitas sendiri dengan IP atau informasi hostname.

Izin ini diperiksa ketika perangkat core Greengrass mencoba untuk memperbarui informasi konektivitasnya di cloud.

`greengrass:UpdateCoreDeploymentStatus`

Izin untuk perangkat core Greengrass untuk memperbarui status deployment.

Izin ini diperiksa ketika perangkat core Greengrass menerima deployment lalu mencoba untuk memperbarui status deployment.

Tindakan Perangkat Greengrass

AWS IoT Greengrassmendefinisikan tindakan kebijakan berikut yang dapat digunakan perangkat klien dalam AWS IoT kebijakan:

`greengrass:Discover`

Izin untuk perangkat klien untuk menggunakan Discovery API untuk mengambil informasi konektivitas inti grup dan otoritas sertifikat grup.

Izin ini diperiksa ketika perangkat klien memanggil Discovery API dengan otentikasi timbal balik TLS.

Minimal AWS IoT kebijakan untuk AWS IoT Greengrass perangkat core

Kebijakan contoh berikut mencakup serangkaian tindakan minimum yang diperlukan untuk mendukung fungsi Greengrass dasar untuk perangkat core Anda.

- Kebijakan mencantumkan topik MQTT dan filter topik yang dapat diterbitkan oleh perangkat core, berlangganan, dan menerima pesan, termasuk topik yang digunakan untuk status bayangan. Untuk mendukung pertukaran pesan antaraAWS IoT Core, fungsi Lambda, konektor, dan perangkat klien di grup Greengrass, tentukan topik dan filter topik yang ingin Anda izinkan. Untuk informasi lebih lanjut, lihat contoh kebijakan Terbitkan/Berlangganan di AWS IoT Core Panduan Developer.
- Kebijakan ini mencakup bagian yang mengizinkan AWS IoT Core Untuk mendapatkan, memperbarui, dan menghapus bayangan perangkat core. Untuk mengizinkan sinkronisasi bayangan untuk perangkat klien di grup Greengrass, tentukan target Nama Sumber Daya Amazon (ARN) dalam `Resource` daftar (misalnya,). `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`device-name`*
- Penggunaan variabel kebijakan hal (`iot:Connection.Thing.*`) di kebijakan AWS IoT untuk perangkat core tidak didukung. Core menggunakan sertifikat perangkat yang sama untuk membuat banyak koneksi ke AWS IoT Core tetapi ID klien dalam koneksi mungkin tidak sama persis dengan nama core.
- Untuk `greengrass:UpdateCoreDeploymentStatus` izin, segmen akhir dalam `Resource` ARN adalah ARN URL-dikodekan dari perangkat core.

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:client/core-name-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish",
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:topic/$aws/things/core-name-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:topicfilter/$aws/things/core-name-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:GetThingShadow",
                "iot:UpdateThingShadow",
                "iot:DeleteThingShadow"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:thing/core-name-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "greengrass:AssumeRoleForGroup",
```

```
                "greengrass:CreateCertificate"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "greengrass:GetDeployment"
            ],
            "Resource": [
                "arn:aws:greengrass:region:account-id:/greengrass/groups/group-id/
deployments/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "greengrass:GetDeploymentArtifacts"
            ],
            "Resource": [
                "arn:aws:greengrass:region:account-id:/greengrass/groups/group-id/
deployments/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "greengrass:UpdateCoreDeploymentStatus"
            ],
            "Resource": [
                "arn:aws:greengrass:region:account-id:/greengrass/groups/group-id/
deployments/*/cores/arn%3Aaws%3Aiot%3Aregion%3Aaccount-id%3Athing%2Fcore-name"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "greengrass:GetConnectivityInfo",
                "greengrass:UpdateConnectivityInfo"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:thing/core-name-*"
```

```
            ]
        }
    ]
}
```

> **Note**
>
> AWS IoTkebijakan untuk perangkat klien biasanya memerlukan izin serupa untuk`iot:Connect`,, `iot:Publish`, `iot:Receive`, dan `iot:Subscribe` tindakan. Untuk memungkinkan perangkat klien mendeteksi informasi konektivitas secara otomatis untuk inti dalam grup Greengrass tempat perangkat tersebut berada, kebijakan untuk perangkat klien harus menyertakan tindakan AWS IoT tersebut. `greengrass:Discover` Di `Resource` bagian ini, tentukan ARN perangkat klien, bukan ARN dari perangkat inti Greengrass. Sebagai contoh:
>
> ```
> {
>     "Effect": "Allow",
>     "Action": [
>         "greengrass:Discover"
>     ],
>     "Resource": [
>         "arn:aws:iot:region:account-id:thing/device-name"
>     ]
> }
> ```
>
> AWS IoTKebijakan untuk perangkat klien biasanya tidak memerlukan izin untuk`iot:GetThingShadow`,, atau `iot:DeleteThingShadow` tindakan`iot:UpdateThingShadow`, karena inti Greengrass menangani operasi sinkronisasi bayangan untuk perangkat klien. Dalam hal ini, pastikan bahwa `Resource` bagian untuk tindakan bayangan dalam AWS IoT kebijakan inti mencakup ARN perangkat klien.

Di AWS IoT konsol, Anda dapat melihat dan mengedit kebijakan yang dilampirkan ke sertifikat core Anda.

1. Di panel navigasi, di bawah Kelola, perluas Semua perangkat, lalu pilih Things.
2. Pilih inti Anda.

3. Pada halaman konfigurasi inti Anda, pilih tab Sertifikat.
4. Di tab Sertifikat, pilih sertifikat Anda.
5. Pada halaman konfigurasi sertifikat, pilih Kebijakan, lalu pilih kebijakan.

   Jika Anda ingin mengedit kebijakan, pilih Edit versi aktif.
6. Tinjau kebijakan dan tambahkan, hapus, atau edit izin sesuai kebutuhan.
7. Untuk menetapkan versi kebijakan baru sebagai versi aktif, di bawah Status versi Kebijakan, pilih Setel versi yang diedit sebagai versi aktif untuk kebijakan ini.
8. Pilih Simpan sebagai versi baru.

Identity and access management untuk AWS IoT Greengrass

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke sumber daya AWS secara aman. Administrator IAM mengontrol siapa yang dapat terautentikasi (masuk) dan berwenang (memiliki izin) untuk menggunakan sumber daya AWS IoT Greengrass. IAM adalah layanan Layanan AWS yang dapat Anda gunakan tanpa dikenakan biaya tambahan.

> **Note**
>
> Topik ini menjelaskan konsep dan fitur IAM. Untuk informasi tentang fitur IAM yang didukung oleh AWS IoT Greengrass, lihat the section called “Bagaimana AWS IoT Greengrass bekerja dengan IAM”.

Audiens

Cara menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan di AWS IoT Greengrass.

Pengguna layanan – Jika Anda menggunakan layanan AWS IoT Greengrass untuk melakukan tugas Anda, administrator Anda akan memberikan kredensial dan izin yang Anda butuhkan. Saat Anda menggunakan lebih banyak fitur AWS IoT Greengrass untuk melakukan pekerjaan, Anda mungkin memerlukan izin tambahan. Memahami cara mengelola akses dapat membantu Anda meminta izin yang tepat dari administrator Anda. Jika Anda tidak dapat mengakses fitur di AWS IoT Greengrass, lihat Pemecahan masalah identitas dan akses untuk AWS IoT Greengrass.

Administrator layanan – Jika Anda bertanggung jawab atas sumber daya AWS IoT Greengrass di perusahaan Anda, Anda mungkin memiliki akses penuh ke AWS IoT Greengrass. Tugas Anda adalah menentukan AWS IoT Greengrass fitur dan sumber daya mana yang harus diakses pengguna layanan Anda. Kemudian, Anda harus mengirimkan permintaan kepada administrator IAM Anda untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep dasar IAM. Untuk mempelajari lebih lanjut tentang cara perusahaan Anda dapat menggunakan IAM dengan AWS IoT Greengrass, lihat Bagaimana AWS IoT Greengrass bekerja dengan IAM.

Administrator IAM – Jika Anda adalah administrator IAM, Anda mungkin ingin belajar dengan lebih detail tentang cara Anda menulis kebijakan untuk mengelola akses ke AWS IoT Greengrass. Untuk melihat contoh kebijakan berbasis identitas AWS IoT Greengrass yang dapat Anda gunakan di IAM, lihat Contoh kebijakan berbasis identitas untuk AWS IoT Greengrass.

Autentikasi menggunakan identitas

Autentikasi adalah cara Anda untuk masuk ke AWS menggunakan kredensial identitas Anda. Anda harus terautentikasi (masuk ke AWS) sebagai Pengguna root akun AWS, sebagai pengguna IAM, atau dengan mengambil peran IAM.

Anda dapat masuk ke AWS sebagai identitas terfederasi dengan menggunakan kredensial yang disediakan melalui sumber identitas. Pengguna AWS IAM Identity Center Pengguna (Pusat Identitas IAM), autentikasi Single Sign-On perusahaan Anda, dan kredensial Google atau Facebook Anda merupakan contoh identitas terfederasi. Saat Anda masuk sebagai identitas gabungan, administrator Anda sebelumnya menyiapkan federasi identitas menggunakan peran IAM. Ketika Anda mengakses AWS dengan menggunakan federasi, Anda secara tidak langsung mengambil suatu peran.

Bergantung pada jenis pengguna Anda, Anda dapat masuk ke AWS Management Console atau portal akses AWS. Untuk informasi selengkapnya tentang cara masuk ke AWS, lihat Cara masuk ke Akun AWS dalam Panduan Pengguna AWS Sign-In.

Jika Anda mengakses AWS secara terprogram, AWS memberikan Kit Pengembangan Perangkat Lunak (SDK) dan antarmuka baris perintah (CLI) untuk menandatangani permintaan Anda secara kriptografis dengan menggunakan kredensial Anda. Jika Anda tidak menggunakan peralatan AWS, Anda harus menandatangani permintaan sendiri. Untuk informasi selengkapnya tentang cara menggunakan metode yang disarankan untuk menandatangani permintaan sendiri, lihat Menandatangani permintaan API AWS dalam Panduan Pengguna IAM.

Apa pun metode autentikasi yang digunakan, Anda mungkin diminta untuk menyediakan informasi keamanan tambahan. Sebagai contoh, AWS menyarankan Anda menggunakan autentikasi multi-faktor (MFA) untuk meningkatkan keamanan akun Anda. Untuk mempelajari lebih lanjut, lihat Autentikasi multi-faktor dalam Panduan Pengguna AWS IAM Identity Center dan Menggunakan autentikasi multi-faktor (MFA) di AWS dalam Panduan Pengguna IAM.

Pengguna root Akun AWS

Ketika membuat Akun AWS, Anda memulai dengan satu identitas masuk yang memiliki akses penuh ke semua Layanan AWS dan sumber daya di akun tersebut. Identitas ini disebut pengguna root Akun AWS dan diakses dengan cara masuk menggunakan alamat email dan kata sandi yang Anda gunakan untuk membuat akun. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari Anda. Lindungi kredensial pengguna root Anda dan gunakan kredensial tersebut untuk melakukan tugas yang hanya dapat dilakukan pengguna root. Untuk daftar tugas lengkap yang mengharuskan Anda masuk sebagai pengguna root, lihat Tugas yang memerlukan kredensial pengguna root dalam Panduan Pengguna IAM.

Pengguna dan grup IAM

Pengguna IAM adalah identitas dalam Akun AWS Anda yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, sebaiknya andalkan kredensial temporer, dan bukan membuat pengguna IAM yang memiliki kredensial jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan khusus yang memerlukan kredensial jangka panjang dengan pengguna IAM, sebaiknya rotasikan kunci akses. Untuk informasi selengkapnya, lihat Merotasi kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensial jangka panjang dalam Panduan Pengguna IAM.

Grup IAM adalah identitas yang menentukan kumpulan pengguna IAM. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin untuk beberapa pengguna sekaligus. Grup membuat izin lebih mudah dikelola untuk sekelompok besar pengguna. Misalnya, Anda dapat memiliki grup yang bernama IAMAdmins dan memberikan izin kepada grup tersebut untuk mengelola sumber daya IAM.

Pengguna berbeda dari peran. Pengguna secara unik terkait dengan satu orang atau aplikasi, tetapi peran tersebut dimaksudkan untuk dapat diambil oleh siapa pun yang membutuhkannya. Pengguna memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara. Untuk mempelajari selengkapnya, silakan lihat Kapan harus membuat pengguna IAM (bukan peran) dalam Panduan Pengguna IAM.

Peran IAM

Peran IAM merupakan identitas dalam Akun AWS Anda yang memiliki izin khusus. Peran ini mirip dengan pengguna IAM, tetapi tidak terkait dengan orang tertentu. Anda dapat mengambil peran IAM untuk sementara dalam AWS Management Console dengan berganti peran. Anda dapat mengambil peran dengan cara memanggil operasi API AWS CLI atau AWS atau menggunakan URL kustom. Untuk informasi selengkapnya tentang metode untuk menggunakan peran, lihat Menggunakan peran IAM dalam Panduan Pengguna IAM.

Peran IAM dengan kredensial sementara berguna dalam situasi berikut:

- Akses pengguna gabungan – Untuk menetapkan izin ke sebuah identitas gabungan, Anda dapat membuat peran dan menentukan izin untuk peran tersebut. Saat identitas terfederasi diautentikasi, identitas tersebut dikaitkan dengan peran dan diberikan izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat Membuat peran untuk Penyedia Identitas pihak ketiga dalam Panduan Pengguna IAM. Jika Anda menggunakan Pusat Identitas IAM, Anda mengonfigurasi sekumpulan izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah identitas tersebut diautentikasi, Pusat Identitas IAM mengaitkan izin yang ditetapkan ke peran dalam IAM. Untuk informasi tentang rangkaian izin, lihat Rangkaian izin dalam Panduan Pengguna AWS IAM Identity Center.
- Izin pengguna IAM sementara – Pengguna atau peran IAM dapat mengambil peran IAM guna mendapatkan berbagai izin secara sementara untuk tugas tertentu.
- Akses lintas akun – Anda dapat menggunakan peran IAM untuk mengizinkan seseorang (pengguna utama tepercaya) dengan akun berbeda untuk mengakses sumber daya yang ada di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, pada beberapa Layanan AWS, Anda dapat menyertakan kebijakan secara langsung ke sumber daya (bukan menggunakan peran sebagai proksi). Untuk mempelajari perbedaan antara kebijakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Bagaimana peran IAM berbeda dari kebijakan berbasis sumber daya dalam Panduan Pengguna IAM.
- Akses lintas layanan – Sebagian Layanan AWS menggunakan fitur di Layanan AWS lainnya. Contoh, ketika Anda melakukan panggilan dalam layanan, umumnya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Suatu layanan mungkin melakukan hal tersebut menggunakan izin pengguna utama panggilan, menggunakan peran layanan, atau peran terkait layanan.
  - Sesi akses maju (FAS) – Ketika Anda menggunakan pengguna IAM atau peran IAM untuk melakukan tindakan di AWS, Anda akan dianggap sebagai seorang pengguna utama. Saat menggunakan beberapa layanan, Anda mungkin melakukan tindakan yang kemudian dilanjutkan

oleh tindakan lain pada layanan yang berbeda. FAS menggunakan izin dari pengguna utama untuk memanggil Layanan AWS, yang dikombinasikan dengan Layanan AWS yang diminta untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya diajukan saat layanan menerima permintaan yang memerlukan interaksi dengan Layanan AWS lain atau sumber daya lain untuk diselesaikan. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat Meneruskan sesi akses.

- Peran IAM – Peran layanan adalah peran IAM yang diambil layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, memodifikasi, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.
- Peran terkait layanan – Peran terkait layanan adalah tipe peran layanan yang terkait dengan Layanan AWS. Layanan tersebut dapat mengambil peran untuk melakukan sebuah tindakan atas nama Anda. Peran terkait layanan akan muncul di Akun AWS Anda dan dimiliki oleh layanan tersebut. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.

- Aplikasi yang berjalan di Amazon EC2 – Anda dapat menggunakan peran IAM untuk mengelola kredensial sementara untuk aplikasi yang berjalan di instans EC2 dan mengajukan permintaan API AWS CLI atau AWS. Cara ini lebih dianjurkan daripada menyimpan kunci akses dalam instans EC2. Untuk menetapkan peran AWS ke instans EC2 dan menyediakannya bagi semua aplikasinya, Anda dapat membuat profil instans yang dilampirkan ke instans tersebut. Profil instans berisi peran dan memungkinkan program yang berjalan di instans EC2 mendapatkan kredensial sementara. Untuk informasi selengkapnya, lihat Menggunakan peran IAM untuk memberikan izin ke aplikasi yang berjalan di instans Amazon EC2 dalam Panduan Pengguna IAM.

Untuk mempelajari apakah kita harus menggunakan peran IAM atau pengguna IAM, lihat Kapan harus membuat peran IAM (bukan pengguna) dalam Panduan Pengguna IAM.

Mengelola akses menggunakan kebijakan

Anda mengendalikan akses di AWS dengan membuat kebijakan dan melampirkannya ke identitas atau sumber daya AWS. Kebijakan adalah objek di AWS yang, ketika terkait dengan identitas atau sumber daya, akan menentukan izinnya. AWS mengevaluasi kebijakan-kebijakan tersebut ketika seorang pengguna utama (pengguna, pengguna root, atau sesi peran) mengajukan permintaan. Izin dalam kebijakan menentukan apakah permintaan diizinkan atau ditolak. Sebagian besar kebijakan disimpan di AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang struktur dan

isi dokumen kebijakan JSON, silakan lihat Gambaran Umum kebijakan JSON dalam Panduan Pengguna IAM.

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan secara spesifik siapa yang memiliki akses terhadap apa. Artinya, pengguna utama manakah yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Untuk memberikan izin kepada pengguna untuk melakukan tindakan pada sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat menjalankan peran.

Kebijakan IAM mendefinisikan izin untuk suatu tindakan terlepas dari metode yang Anda gunakan untuk operasi. Sebagai contoh, anggap saja Anda memiliki kebijakan yang mengizinkan tindakan `iam:GetRole`. Pengguna dengan kebijakan tersebut dapat memperoleh informasi peran dari AWS Management Console, AWS CLI, atau API AWS.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan pengguna dan peran, di sumber daya mana, dan dengan ketentuan apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan terkelola. Kebijakan inline disematkan langsung ke satu pengguna, grup, atau peran. Kebijakan terkelola adalah kebijakan mandiri yang dapat Anda lampirkan ke beberapa pengguna, grup, dan peran di Akun AWS Anda. Kebijakan terkelola meliputi kebijakan yang dikelola AWS dan kebijakan yang dikelola pelanggan. Untuk mempelajari cara memilih antara kebijakan terkelola atau kebijakan inline, lihat Memilih antara kebijakan terkelola dan kebijakan inline dalam Panduan Pengguna IAM.

Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya yang dilampiri kebijakan tersebut, kebijakan ini menentukan jenis tindakan yang

dapat dilakukan oleh pengguna utama tertentu di sumber daya tersebut dan apa ketentuannya. Anda harus menentukan pengguna utama dalam kebijakan berbasis sumber daya. Pengguna utama dapat mencakup akun, pengguna, peran, pengguna gabungan, atau Layanan AWS.

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan yang dikelola AWS dari IAM dalam kebijakan berbasis sumber daya.

Daftar kontrol akses (ACL)

Daftar kontrol akses (ACL) mengendalikan pengguna utama mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL serupa dengan kebijakan berbasis sumber daya, meskipun tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung ACL. Untuk mempelajari ACL selengkapnya, silakan lihat Gambaran umum daftar kontrol akses (ACL) di Panduan Developer Layanan Penyimpanan Ringkas Amazon.

Tipe kebijakan lain

AWS mendukung jenis kebijakan tambahan yang kurang umum. Tipe-tipe kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda berdasarkan tipe kebijakan yang lebih umum.

- Batasan izin – Batasan izin adalah fitur lanjutan di mana Anda menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas IAM (pengguna atau peran IAM). Anda dapat menetapkan batasan izin untuk suatu entitas. Izin yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas milik entitas dan batasan izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan secara eksplisit terhadap salah satu kebijakan ini akan mengesampingkan izin tersebut. Untuk informasi selengkapnya tentang batasan izin, lihat Batasan izin untuk entitas IAM dalam Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCP) – SCP adalah kebijakan JSON yang menentukan izin maksimum untuk sebuah organisasi atau unit organisasi (OU) di AWS Organizations. AWS Organizations adalah layanan untuk mengelompokkan dan mengelola beberapa akun Akun AWS yang dimiliki bisnis Anda secara terpusat. Jika Anda mengaktifkan semua fitur di organisasi, Anda dapat menerapkan kebijakan kontrol layanan (SCP) ke salah satu atau semua akun Anda. SCP membatasi izin untuk entitas dalam akun anggota, termasuk setiap Pengguna root akun AWS. Untuk informasi selengkapnya tentang Organisasi dan SCP, lihat Cara kerja SCP dalam Panduan Pengguna AWS Organizations.

- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda teruskan sebagai parameter saat Anda membuat sesi sementara secara terprogram untuk peran atau pengguna gabungan. Izin sesi yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga bisa datang dari kebijakan berbasis sumber daya. Penolakan eksplisit di salah satu kebijakan ini akan membatalkan izin tersebut. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

Berbagai jenis kebijakan

Jika beberapa jenis kebijakan diberlakukan untuk satu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari bagaimana AWS menentukan apakah akan mengizinkan permintaan atau tidak jika beberapa jenis kebijakan terlibat, lihat Logika evaluasi kebijakan dalam Panduan Pengguna IAM.

Lihat juga

- the section called “Bagaimana AWS IoT Greengrass bekerja dengan IAM”
- the section called “Contoh kebijakan berbasis identitas”
- the section called “Pemecahan masalah identitas dan akses”

Bagaimana AWS IoT Greengrass bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses ke AWS IoT Greengrass, Anda harus memahami fitur IAM apa yang tersedia untuk digunakan dengan AWS IoT Greengrass.

Fitur IAM	Didukung oleh Greengrass?
Kebijakan berbasis identitas dengan izin tingkat sumber daya	Ya
Kebijakan berbasis sumber daya	Tidak
Daftar kontrol akses (ACL)	Tidak
Otorisasi berbasis tag	Ya
Kredensial sementara	Ya

Fitur IAM	Didukung oleh Greengrass?
Peran terkait layanan	Tidak
Peran layanan	Ya

Untuk mendapatkan tampilan tingkat tinggi tentang cara dan layanan AWS lainnya bekerja dengan IAM, lihat AWS layanan yang bekerja dengan IAM dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas untuk AWS IoT Greengrass

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan tindakan dan sumber daya yang diizinkan atau ditolak, dan juga ketentuan di mana tindakan tersebut diperbolehkan atau ditolak. AWS IoT Greengrass mendukung tindakan, sumber daya, dan kunci kondisi tertentu. Untuk mempelajari semua elemen yang Anda gunakan dalam kebijakan JSON, lihat Referensi elemen kebijakan IAM JSON dalam Panduan Pengguna IAM.

Tindakan

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan siapa yang memiliki akses ke apa. Yaitu, pengguna utama mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Elemen `Action` dari kebijakan JSON menjelaskan tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan. Tindakan kebijakan biasanya memiliki nama yang sama seperti operasi API AWS terkait. Ada beberapa pengecualian, misalnya tindakan hanya izin yang tidak memiliki operasi API yang cocok. Ada juga beberapa operasi yang memerlukan beberapa tindakan dalam suatu kebijakan. Tindakan tambahan ini disebut tindakan dependen.

Menyertakan tindakan dalam suatu kebijakan untuk memberikan izin melakukan operasi terkait.

Tindakan kebijakan AWS IoT Greengrass menggunakan `greengrass:` prefiks berikut sebelum tindakan:. Sebagai contoh, untuk mengizinkan seseorang untuk menggunakan `ListGroups` operasi API untuk daftar grup di mereka Akun AWS, Anda termasuk `greengrass:ListGroups` tindakan dalam kebijakan mereka. Pernyataan kebijakan harus mencakup salah satu `Action` atau `NotAction` elemen. AWS IoT Greengrass mendefinisikan serangkaian tindakan yang menjelaskan tugas yang Anda dapat lakukan dengan layanan ini.

Untuk menetapkan beberapa tindakan dalam satu pernyataan, letakkan dalam tanda kurung (`[ ]`) dan pisahkan dengan koma seperti berikut:

```
"Action": [
    "greengrass:action1",
    "greengrass:action2",
    "greengrass:action3"
]
```

Anda bisa menggunakan wildcard (*) untuk menentukan beberapa tindakan. Sebagai contoh, untuk menentukan semua tindakan yang dimulai dengan kata `List`, sertakan tindakan berikut:

```
"Action": "greengrass:List*"
```

> **Note**
>
> Kami merekomendasikan Anda menghindari penggunaan wildcard untuk menentukan semua tindakan yang tersedia untuk layanan. Sebagai praktik terbaik, Anda harus memberi setidaknya hak istimewa dan izin cakupan secara sempit dalam kebijakan. Untuk informasi selengkapnya, lihat the section called “Berikan izin minimum yang memungkinkan”.

Untuk melihat daftar tindakan AWS IoT Greengrass, lihat Tindakan yang Ditetapkan oleh AWS IoT Greengrass di Panduan Pengguna IAM.

Sumber daya

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan siapa yang memiliki akses ke apa. Yaitu, pengguna utama mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Elemen kebijakan JSON `Resource` menentukan objek atau beberapa objek yang menjadi target penerapan tindakan. Pernyataan harus menyertakan elemen `Resource` atau `NotResource`. Praktik terbaiknya, tentukan sumber daya menggunakan Amazon Resource Name (ARN). Anda dapat melakukan ini untuk tindakan yang mendukung jenis sumber daya tertentu, yang dikenal sebagai izin tingkat sumber daya.

Untuk tindakan yang tidak mendukung izin tingkat sumber daya, misalnya operasi pembuatan daftar, gunakan wildcard (*) untuk menunjukkan bahwa pernyataan tersebut berlaku bagi semua sumber daya.

```
"Resource": "*"
```

Tabel berikut berisi AWS IoT Greengrass sumber daya ARN yang dapat digunakan dalam `Resource` elemen pernyataan kebijakan. Untuk pemetaan izin tingkat sumber daya yang didukung untuk AWS IoT Greengrass tindakan, lihat Tindakan Ditetapkan oleh AWS IoT Greengrass dalam Panduan Pengguna IAM.

Sumber daya	ARN
Group	`arn:${Partition}:greengrass:${Region}:${Account}:/greengrass/groups/${GroupId}`
GroupVersion	`arn:${Partition}:greengrass:${Region}:${Account}:/greengrass/groups/${GroupId}/versions/${VersionId}`
CertificateAuthority	`arn:${Partition}:greengrass:${Region}:${Account}:/greengrass/groups/${GroupId}/certificateauthorities/${CertificateAuthorityId}`
Deployment	`arn:${Partition}:greengrass:${Region}:${Account}:/greengrass/groups/${GroupId}/deployments/${DeploymentId}`
BulkDeployment	`arn:${Partition}:greengrass:${Region}:${Account}:/greengrass/bulk/deployments/${BulkDeploymentId}`
ConnectorDefinition	`arn:${Partition}:greengrass:${Region}:${Account}:/greengrass/definition/connectors/${ConnectorDefinitionId}`
ConnectorDefinitionVersion	`arn:${Partition}:greengrass:${Region}:${Account}:/greengrass/definition/connectors/${ConnectorDefinitionId}/versions/${VersionId}`
CoreDefinition	`arn:${Partition}:greengrass:${Region}:${Account}:/greengrass/definition/cores/${CoreDefinitionId}`
CoreDefinitionVersion	`arn:${Partition}:greengrass:${Region}:${Account}:/greengrass/definition/cores/${CoreDefinitionId}/versions/${VersionId}`

Sumber daya	ARN
DeviceDefinition	arn:${Partition}:greengrass:${Region}:${Account}:/greengrass/definition/devices/${DeviceDefinitionId}
DeviceDefinitionVersion	arn:${Partition}:greengrass:${Region}:${Account}:/greengrass/definition/devices/${DeviceDefinitionId}/versions/${VersionId}
FunctionDefinition	arn:${Partition}:greengrass:${Region}:${Account}:/greengrass/definition/functions/${FunctionDefinitionId}
FunctionDefinitionVersion	arn:${Partition}:greengrass:${Region}:${Account}:/greengrass/definition/functions/${FunctionDefinitionId}/versions/${VersionId}
LoggerDefinition	arn:${Partition}:greengrass:${Region}:${Account}:/greengrass/definition/loggers/${LoggerDefinitionId}
LoggerDefinitionVersion	arn:${Partition}:greengrass:${Region}:${Account}:/greengrass/definition/loggers/${LoggerDefinitionId}/versions/${VersionId}
ResourceDefinition	arn:${Partition}:greengrass:${Region}:${Account}:/greengrass/definition/resources/${ResourceDefinitionId}
ResourceDefinitionVersion	arn:${Partition}:greengrass:${Region}:${Account}:/greengrass/definition/resources/${ResourceDefinitionId}/versions/${VersionId}
SubscriptionDefinition	arn:${Partition}:greengrass:${Region}:${Account}:/greengrass/definition/subscriptions/${SubscriptionDefinitionId}

Sumber daya	ARN
SubscriptionDefinitionVersion	`arn:${Partition}:greengrass:${Region}:${Account}:/greengrass/definition/subscriptions/${SubscriptionDefinitionId}/versions/${VersionId}`
ConnectivityInfo	`arn:${Partition}:greengrass:${Region}:${Account}:/greengrass/things/${ThingName}/connectivityInfo`

Contoh berikut `Resource` slemen menentukan ARN dari grup di Wilayah US West (Oregon) di Akun AWS `123456789012`:

```
"Resource": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/groups/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

Atau, untuk menentukan semua grup yang dimiliki Akun AWS dalam spesifik Wilayah AWS, gunakan wildcard di tempat ID grup:

```
"Resource": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/groups/*"
```

Beberapa AWS IoT Greengrass tindakan (sebagai contoh, beberapa operasi daftar), tidak dapat dilakukan pada sumber daya tertentu. Dalam kondisi tersebut, Anda harus menggunakan karakter wildcard saja.

```
"Resource": "*"
```

Untuk menetapkan beberapa ARN sumber daya dalam satu pernyataan, letakkan dalam tanda kurung (`[ ]`) dan pisahkan dengan koma seperti berikut:

```
"Resource": [
    "resource-arn1",
    "resource-arn2",
    "resource-arn3"
]
```

Untuk informasi selengkapnya tentang format ARN, lihat Amazon Resource Names (ARN) dan ruang nama AWS layanan di. Referensi Umum Amazon Web Services

Kunci syarat

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan siapa yang memiliki akses ke hal apa. Yaitu, pengguna utama mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Elemen `Condition` (atau blok `Condition`) memungkinkan Anda menentukan kondisi di mana suatu pernyataan akan diterapkan. Elemen `Condition` bersifat opsional. Anda dapat membuat ekspresi kondisional yang menggunakan operator kondisi, misalnya sama dengan atau kurang dari, untuk mencocokkan kondisi dalam kebijakan dengan nilai-nilai yang diminta.

Jika Anda menentukan beberapa elemen `Condition` dalam satu pernyataan, atau beberapa kunci dalam satu elemen `Condition`, AWS akan mengevaluasinya dengan menggunakan operasi `AND` logis. Jika Anda menentukan beberapa niilai untuk satu kunci persyaratan, AWS akan mengevaluasi syarat tersebut menggunakan operasi `OR` yang logis. Semua kondisi harus dipenuhi sebelum izin pernyataan diberikan.

Anda juga dapat menggunakan variabel placeholder saat menentukan kondisi. Sebagai contoh, Anda dapat memberikan izin kepada pengguna IAM untuk mengakses sumber daya hanya jika izin tersebut mempunyai tanda yang sesuai dengan nama pengguna IAM mereka. Untuk informasi selengkapnya, silakan lihat Elemen kebijakan IAM: variabel dan tanda di Panduan Pengguna IAM.

AWS mendukung kunci kondisi global dan kunci kondisi spesifik layanan. Untuk melihat semua AWS kunci syarat global, lihat AWS kunci konteks syarat global dalam Panduan Pengguna IAM.

AWS IoT Greengrass mendukung kunci syarat global berikut.

Kunci	Deskripsi
`aws:CurrentTime`	Filter mengakses dengan memeriksa kondisi tanggal/waktu untuk tanggal dan waktu saat ini.
`aws:EpochTime`	Filter mengakses dengan memeriksa kondisi tanggal/waktu untuk tanggal dan waktu saat ini dalam jangka waktu atau waktu Unix.
`aws:MultiFactorAuthAge`	Filter mengakses dengan memeriksa berapa lama yang lalu (dalam detik) kredensial keamanan yang divalidasi oleh autentikasi multi faktor (MFA) dalam permintaan dikeluarkan menggunakan MFA.

Kunci	Deskripsi
`aws:MultiFactorAuthPresent`	Filter mengakses ini untuk memeriksa apakah autentikasi multi-faktor (MFA) digunakan untuk memvalidasi kredensial keamanan sementara yang membuat permintaan.
`aws:RequestTag/${TagKey}`	Filter membuat permintaan berdasarkan set yang diizinkan nilai untuk masing-masing tag wajib.
`aws:ResourceTag/${TagKey}`	Filter tindakan berdasarkan menandai nilai yang terkait dengan sumber daya.
`aws:SecureTransport`	Filter mengakses dengan memeriksa apakah permintaan dikirim menggunakan SSL.
`aws:TagKeys`	Filter membuat permintaan berdasarkan adanya tag wajib dalam permintaan.
`aws:UserAgent`	Filter mengakses oleh aplikasi klien pemohon.

Untuk informasi lebih lanjut, lihat kunci konteks syarat global AWS dalam Panduan Pengguna IAM.

Contoh-contoh

Untuk melihat contoh AWS IoT Greengrass kebijakan berbasis identitas, lihat the section called "Contoh kebijakan berbasis identitas".

Kebijakan berbasis sumber daya untuk AWS IoT Greengrass

AWS IoT Greengrass tidak mendukung kebijakan berbasis sumber daya.

Daftar kontrol akses (ACL)

AWS IoT Greengrass tidak mendukung ACL.

Otorisasi berdasarkan tanda AWS IoT Greengrass

Anda dapat melampirkan tag di sumber daya AWS IoT Greengrass atau meneruskan tag dalam permintaan ke AWS IoT Greengrass. Untuk mengendalikan akses berdasarkan tanda, Anda dapat

memberikan informasi tentang tanda di Elemen syarat kebijakan dengan menggunakan kunci syarat `aws:ResourceTag/${TagKey}`, `aws:RequestTag/${TagKey}`, atau `aws:TagKeys`. Untuk informasi selengkapnya, lihat Menandai sumber daya Greengrass Anda.

IAM role untuk AWS IoT Greengrass

IAM role adalah entitas dalam Akun AWS Anda yang memiliki izin khusus.

Menggunakan kredensial sementara dengan AWS IoT Greengrass

Kredensial sementara digunakan untuk masuk bersama gabungan, menjalankan IAM role, atau menjalankan peran lintas-akun. Anda memperoleh kredensyal keamanan sementara dengan memanggil operasi AWS STS API seperti AssumeRoleatau. GetFederationToken

Pada core Greengrass, kredensial sementara untuk peran grup dibuat tersedia untuk fungsi dan konektor Lambda yang ditentukan pengguna. Jika fungsi Lambda Anda menggunakan AWS SDK, Anda tidak perlu menambahkan logika untuk mendapatkan kredensial karena AWS SDK melakukan ini untuk Anda.

Peran terkait layanan

AWS IoT Greengrass tidak mendukung peran terkait layanan.

Peran layanan

Fitur ini memungkinkan layanan untuk menerima peran layanan atas nama Anda. Peran ini mengizinkan layanan untuk mengakses sumber daya di layanan lain untuk menyelesaikan tindakan atas nama Anda. Peran layanan muncul di akun IAM Anda dan dimiliki oleh akun tersebut. Ini berarti administrator IAM dapat mengubah izin untuk peran ini. Namun, melakukan hal itu dapat merusak fungsionalitas layanan.

AWS IoT Greengrass menggunakan peran layanan untuk mengakses beberapa perangkat AWS sumber daya atas nama Anda. Untuk informasi selengkapnya, lihat the section called “Peran layanan Greengrass”.

Memilih peran IAM di konsol AWS IoT Greengrass

Di AWS IoT Greengrass konsol, Anda mungkin harus memilih peran layanan Greengrass atau peran grup Greengrass dari daftar IAM role di akun Anda.

- Peran layanan Greengrass mengizinkan AWS IoT Greengrass untuk mengakses sumber daya AWS Anda di layanan lain atas nama Anda. Biasanya, Anda tidak perlu memilih peran layanan

karena konsol dapat membuat dan mengonfigurasinya untuk Anda. Untuk informasi selengkapnya, lihat the section called “Peran layanan Greengrass”.

- Peran grup Greengrass digunakan untuk mengizinkan fungsi Greengrass Lambda dan konektor dalam grup untuk mengakses Anda AWS sumber daya. Itu juga dapat memberikan AWS IoT Greengrass izin untuk mengekspor aliran ke AWS layanan dan menulis CloudWatch log. Lihat informasi yang lebih lengkap di the section called “Peran grup Greengrass”.

Peran layanan Greengrass

Peran layanan Greengrass adalah AWS Identity and Access Management (IAM) peran layanan yang mengotorisasi AWS IoT Greengrass untuk mengakses sumber daya dari layanan AWS atas nama Anda. Hal ini mengizinkan untuk AWS IoT Greengrass untuk melakukan tugas penting, seperti mengambil fungsi AWS Lambda dan mengelola AWS IoT bayangan Anda.

Untuk mengizinkan AWS IoT Greengrass untuk mengakses sumber daya Anda, peran layanan Greengrass harus dikaitkan dengan Anda Akun AWS dan menentukan AWS IoT Greengrass sebagai entitas terpercaya. Peran harus mencakup AWSGreengrassResourceAccessRolePolicykebijakan terkelola atau kebijakan kustom yang menentukan izin setara untukAWS IoT Greengrassfitur yang Anda gunakan. Kebijakan ini dikelola oleh AWS dan mendefinisikan set izin yang AWS IoT Greengrass digunakan untuk mengakses AWS sumber daya Anda.

Anda dapat menggunakan kembali peran layanan Greengrass yang sama di seluruh Wilayah AWSs, tetapi Anda harus mengaitkannya dengan akun Anda di setiap Wilayah AWS di mana Anda gunakan AWS IoT Greengrass. Deployment grup gagal jika peran layanan tidak ada di saat ini Akun AWS dan Wilayah.

Bagian berikut menjelaskan cara membuat dan mengelola peran layanan Greengrass di AWS Management Console atau AWS CLI.

- Mengelola peran layanan (konsol)
- Mengelola peran layanan (CLI)

> **Note**
>
> Selain peran layanan yang mengesahkan akses tingkat layanan, Anda dapat menugaskan peran grup ke AWS IoT Greengrass grup. Peran grup adalah IAM role terpisah yang

mengontrol cara fungsi Greengrass Lambda dan konektor dalam grup dapat mengakses AWS layanan.

Mengelola peran layanan Greengrass (konsol)

Konsol AWS IoT membuatnya mudah untuk mengelola peran layanan Greengrass Anda. Sebagai contoh, ketika Anda membuat atau men-deploy grup Greengrass, konsol memeriksa apakah Akun AWS terlampir pada peran layanan Greengrass di Wilayah AWS yang ketika ini terpilih di konsol. Jika tidak, konsol dapat membuat dan mengonfigurasi peran layanan untuk Anda. Untuk informasi selengkapnya, lihat the section called “Buat peran layanan Greengrass”.

Anda dapat menggunakan AWS IoT konsol untuk tugas-tugas manajemen peran berikut:

- Temukan peran layanan Greengrass Anda
- Buat peran layanan Greengrass
- Ubah peran layanan Greengrass
- Melepaskan peran layanan Greengrass

> **Note**
>
> Pengguna yang masuk ke konsol harus memiliki izin untuk melihat, membuat, atau mengubah peran layanan.

Temukan peran layanan Greengrass Anda (konsol)

Gunakan langkah-langkah berikut untuk menemukan peran layanan yang AWS IoT Greengrass digunakan saat ini Wilayah AWS.

1. DariAWS IoTkonsolpanel navigasi, pilihPengaturan.
2. Gulir ke Peran layanan Greengrass untuk melihat peran layanan dan kebijakannya.

   Jika Anda tidak melihat peran layanan, Anda dapat membiarkan konsol membuat atau mengonfigurasinya untuk Anda. Untuk informasi selengkapnya, lihat Buat peran layanan Greengrass.

Buat peran layanan Greengrass (konsol)

Konsol dapat membuat dan mengkonfigurasi peran layanan Greengrass default untuk Anda. Peran ini memiliki properti berikut.

Properti	Nilai
Nama	`Greengrass_ServiceRole`
Entitas tepercaya	`AWS service: greengrass`
Kebijakan	AWSGreengrassResourceAccessRolePolicy

> **Note**
>
> Jika Penyiapan perangkat Greengrass membuat peran layanan, nama perannya adalah `GreengrassServiceRole_`*`random-string`*.

Ketika Anda membuat atau men-deploy grup Greengrass dari konsol AWS IoT tersebut, konsol tersebut memeriksa apakah peran layanan Greengrass dikaitkan dengan Akun AWS Anda di Wilayah AWS yang mana terpilih di konsol. Jika tidak, konsol akan meminta Anda untuk mengizinkan AWS IoT Greengrass untuk membaca dan menulis ke AWS layanan atas nama Anda.

Jika Anda memberikan izin, konsol tersebut akan memeriksa apakah peran bernama `Greengrass_ServiceRole` ada di Akun AWS Anda.

- Jika peran itu ada, konsol tersebut akan melampirkan peran layanan ke perangkat Akun AWS di Wilayah AWS saat ini.
- Jika peran tersebut tidak ada, konsol tersebut akan membuat peran layanan Greengrass default dan melampirkannya ke Akun AWS Anda di Wilayah AWS saat ini.

> **Note**
>
> Jika Anda ingin membuat peran layanan dengan kebijakan peran kustom, gunakan konsol IAM untuk membuat atau mengubah peran. Untuk informasi selengkapnya,

lihat Membuat peran untuk mendelegasikan izin ke layanan AWS atau Mengubah peran dalam Panduan Pengguna IAM. Pastikan bahwa peran memberikan izin yang setara dengan kebijakan terkelola `AWSGreengrassResourceAccessRolePolicy` untuk fitur dan sumber daya yang Anda gunakan. Kami merekomendasikan Anda juga menyertakan`aws:SourceArn`dan`aws:SourceAccount`kunci konteks kondisi global dalam kebijakan kepercayaan Anda untuk membantu mencegahdeputi bingungmasalah keamanan. Kunci konteks kondisi membatasi akses untuk mengizinkan hanya permintaan yang berasal dari akun tertentu dan ruang kerja Greengrass. Untuk informasi lebih lanjut tentang masalah deputi yang membingungkan, lihatCross-service bingung wakil pencegahan.
Jika Anda membuat peran layanan, kembali ke AWS IoT konsol dan melampirkan peran ke grup. Anda dapat melakukannya di bawah peran layanan Greengrass di halaman Pengaturan grup.

Ubah peran layanan Greengrass (konsol)

Gunakan prosedur berikut untuk memilih peran layanan Greengrass yang berbeda untuk dilampirkan ke Akun AWS Anda di Wilayah AWSyang saat ini dipilih di konsol tersebut.

1. DariAWS IoTkonsolpanel navigasi, pilihPengaturan.
2. Di bawah Peran layanan Greengrass, pilih Ubah peran.

   Kotak dialog Perbarui peran layanan Greengrass terbuka dan menunjukkan peran IAM di Akun AWS yang menentukan AWS IoT Greengrass sebagai entitas terpercaya.
3. Pilih peran layanan Greengrass untuk dilampirkan.
4. Pilih Lampirkan peran.

> **Note**
>
> Untuk mengizinkan konsol untuk membuat peran layanan Greengrass default untuk Anda, pilih Buat peran untuk saya sebagai ganti memilih peran dari daftar. Tautan Buat peran untuk saya tidak muncul jika role bernama `Greengrass_ServiceRole` ada di Akun AWS.

Melepaskan peran layanan Greengrass (konsol)

Gunakan prosedur berikut untuk melepaskan peran Greengrass dari Akun AWS di Wilayah AWS saat ini yang terpilih di konsol. Ini mencabut izin untuk AWS IoT Greengrass mengakses AWS layanan di saat ini Wilayah AWS.

> **Important**
>
> Melepaskan peran layanan dapat mengganggu operasi aktif.

1. DariAWS IoTkonsolpanel navigasi, pilihPengaturan.
2. Di bawah Peran layanan Greengrass, pilih Lepaskan peran.
3. Dalam kotak dialog konfirmasi, pilih Lepaskan.

> **Note**
>
> Jika Anda tidak lagi memerlukan peran tersebut, Anda dapat menghapusnya di konsol IAM. Untuk informasi lebih lanjut, lihat Menghapus peran atau profil instans dalam Panduan Pengguna IAM.
> Peran lain mungkin mengizinkan AWS IoT Greengrass untuk mengakses sumber daya Anda. Untuk menemukan semua peran yang mengizinkan AWS IoT Greengrass untuk mengasumsi izin atas nama Anda, di konsol IAM, pada halaman Peran tersebut, cari peran yang mencakup AWS layanan: Greengrass di kolom entitas Terpercaya ini.

Mengelola peran layanan Greengrass (CLI)

Dalam prosedur berikut, kita berasumsi bahwa AWS CLI diinstal dan dikonfigurasi untuk menggunakan perangkat Anda Akun AWS ID. Untuk informasi lebih lanjut, lihat Menginstal AWS baris perintah antarmuka dan Mengonfigurasi AWS CLI dalam AWS Command Line Interface Panduan Pengguna.

Anda dapat menggunakan AWS CLI untuk tugas manajemen peran berikut:

- Dapatkan peran layanan Greengrass
- Buat peran layanan Greengrass
- Hapus peran layanan Greengrass

Dapatkan peran layanan Greengrass (CLI)

Gunakan prosedur berikut untuk mengetahui apakah peran layanan Greengrass dikaitkan dengan Akun AWS Anda di Wilayah AWS.

- Dapatkan peran layanan. Ganti *wilayah* dengan Wilayah AWS Anda (sebagai contoh, `us-west-2`).

  ```
  aws Greengrass get-service-role-for-account --region region
  ```

  Jika peran layanan Greengrass telah dikaitkan dengan akun Anda, metadata peran berikut akan dikembalikan.

  ```
  {
    "AssociatedAt": "timestamp",
    "RoleArn": "arn:aws:iam::account-id:role/path/role-name"
  }
  ```

  Jika tidak ada metadata peran yang dikembalikan, maka Anda harus membuat peran layanan (jika tidak ada) dan mengaitkannya dengan akun di Wilayah AWS.

Buat peran layanan Greengrass (CLI)

Gunakan langkah-langkah berikut untuk membuat peran dan mengaitkannya dengan perangkat Akun AWS Anda.

Untuk membuat peran layanan dengan menggunakan IAM.

1. Buat peran dengan kebijakan kepercayaan yang memungkinkan AWS IoT Greengrass untuk menjalankan peran tersebut. Contoh ini menciptakan peran bernama `Greengrass_ServiceRole`, tetapi Anda dapat menggunakan nama yang berbeda. Kami merekomendasikan Anda juga menyertakan`aws:SourceArn`dan`aws:SourceAccount`kunci konteks kondisi global dalam kebijakan kepercayaan Anda untuk membantu mencegah*deputi bingung*masalah keamanan. Kunci konteks kondisi membatasi akses untuk mengizinkan hanya permintaan yang berasal dari akun tertentu dan ruang kerja Greengrass. Untuk informasi

lebih lanjut tentang masalah deputi yang membingungkan, lihatCross-service bingung wakil pencegahan.

Linux, macOS, or Unix

```
aws iam create-role --role-name Greengrass_ServiceRole --assume-role-policy-
document '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "greengrass.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "account-id"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:greengrass:region:account-id:*"
        }
      }
    }
  ]
}'
```

Windows command prompt

```
aws iam create-role --role-name Greengrass_ServiceRole --assume-role-
policy-document "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect
\":\"Allow\",\"Principal\":{\"Service\":\"greengrass.amazonaws.com\"},
\"Action\":\"sts:AssumeRole\",\"Condition\":{\"ArnLike\":{\"aws:SourceArn
\":\"arn:aws:greengrass:region:account-id:*\"},\"StringEquals\":
{\"aws:SourceAccount\":\"account-id\"}}}]}"
```

2. Salin peran ARN dari metadata peran dalam output. Anda menggunakan ARN untuk mengasosiasikan peran dengan akun Anda.
3. Lampirkan kebijakan `AWSGreengrassResourceAccessRolePolicy` pada peran tersebut.

```
aws iam attach-role-policy --role-name Greengrass_ServiceRole --policy-arn
 arn:aws:iam::aws:policy/service-role/AWSGreengrassResourceAccessRolePolicy
```

Untuk mengaitkan peran layanan dengan Akun AWS Anda

- Hubungkan peran itu dengan akun Anda. Ganti *role-arn* dengan peran layanan ARN dan *wilayah* dengan Wilayah AWS Anda (sebagai contoh, `us-west-2`).

  ```
  aws greengrass associate-service-role-to-account --role-arn role-arn --
  region region
  ```

  Jika berhasil, respons berikut dikembalikan.

  ```
  {
    "AssociatedAt": "timestamp"
  }
  ```

Menghapus peran layanan Greengrass (CLI)

Gunakan langkah-langkah berikut untuk memisahkan peran layanan Greengrass dari perangkat Akun AWS.

- Lepaskan peran layanan dari akun Anda. Ganti *wilayah* dengan Wilayah AWS Anda (misalnya, `us-west-2`).

  ```
  aws greengrass disassociate-service-role-from-account --region region
  ```

  Jika berhasil, respon berikut dikembalikan.

  ```
  {
    "DisassociatedAt": "timestamp"
  }
  ```

> **Note**
>
> Anda harus menghapus peran layanan jika Anda tidak menggunakannya dalam semua Wilayah AWS. Pertama gunakan delete-role-policy untuk melepaskan `AWSGreengrassResourceAccessRolePolicy` kebijakan terkelola dari peran, dan kemudian gunakan delete-role untuk menghapus peran. Untuk informasi lebih lanjut, lihat Menghapus peran atau profil instans dalam Panduan Pengguna IAM.

Lihat juga

- Membuat peran untuk mendelegasikan izin ke layanan AWS di Panduan Pengguna IAM.
- Mengubah peran di Panduan Pengguna IAM
- Menghapus peran atau profil instans dalam Panduan Pengguna IAM.
- Perintah AWS IoT Greengrass di Referensi Perintah AWS CLI
  - associate-service-role-toakun
  - disassociate-service-role-fromakun
  - get-service-role-forakun
- Perintah IAM di Referensi PerintahAWS CLI
  - attach-role-policy
  - create-role
  - delete-role
  - delete-role-policy

Peran grup Greengrass

Peran grup Greengrass adalah IAM role yang mengotorisasi kode yang berjalan pada core Greengrass untuk mengakses AWS sumber daya. Anda membuat peran dan mengelola izin di AWS Identity and Access Management (IAM) dan melampirkan peran untuk grup Greengrass Anda. Sebuah grup Greengrass memiliki satu peran grup. Untuk menambah atau mengubah izin, Anda dapat melampirkan peran yang berbeda atau mengubah kebijakan IAM yang dilampirkan ke peran.

Peran harus mendefinisikan AWS IoT Greengrass sebagai entitas terpercaya. Tergantung pada kasus bisnis Anda, peran grup mungkin berisi kebijakan IAM yang menentukan:

- Izin untuk fungsi Lambda yang ditentukan penggunas untuk mengakses AWS layanan.
- Izin untuk konektor untuk mengakses AWS layanan.
- Izin untuk pengelola aliran untuk mengekspor aliran ke AWS IoT Analytics dan Kinesis Data Streams.
- Izin untuk mengizinkanCloudWatch penebangan.

Bagian berikut menjelaskan cara melampirkan atau melepaskan peran grup Greengrass di AWS Management Console atau AWS CLI.

- Mengelola peran grup (konsol)
- Kelola peran grup (CLI)

> **Note**
>
> Selain peran grup yang mengotorisasi akses dari core Greengrass, Anda dapat menetapkan Gperan layanan Greengrasse yang mengizinkan untuk AWS IoT Greengrass untuk mengakses AWS sumber daya atas nama Anda.

Mengelola peran grup Greengrass (konsol)

Anda dapat menggunakan konsol AWS IoT untuk tugas-tugas manajemen peran berikut:

- Temukan peran grup Greengrass Anda
- Tambahkan atau ubah peran grup Greengrass
- Hapus peran grup Greengrass

> **Note**
>
> Pengguna yang masuk ke konsol harus memiliki izin untuk mengelola peran.

Temukan peran grup Greengrass Anda (konsol)

Ikuti langkah-langkah untuk menemukan peran yang melekat pada grup Greengrass.

1. DiAWS IoTpanel navigasi konsol, di bawahKelola, PerluasPerangkat Greengrass, dan kemudian pilihGrup (V1).
2. Pilih grup target.
3. Pada halaman konfigurasi grup, pilihMelihat pengaturan.

Jika peran dilampirkan ke grup, itu muncul di bawahPeran grup.

Tambahkan atau ubah peran grup Greengrass (konsol)

Ikuti langkah-langkah berikut untuk memilih IAM role dari Akun AWS untuk menambahkan ke grup Greengrass.

Peran grup memiliki persyaratan sebagai berikut:

- AWS IoT Greengrass didefinisikan sebagai entitas terpercaya.
- Kebijakan izin yang dilamprkan ke peran harus memberikan izin ke AWS sumber daya yang diperlukan oleh fungsi Lambda dan konektor dalam grup, dan oleh komponen sistem Greengrass.

> **Note**
>
> Kami menyarankan Anda juga memasukkan`aws:SourceArn`dan`aws:SourceAccount`kunci konteks kondisi global dalam kebijakan kepercayaan Anda untuk membantu mencegahdeputyMasalah keamanan. Kunci konteks kondisi membatasi akses untuk mengizinkan hanya permintaan yang berasal dari akun tertentu dan ruang kerja Greengrass. Untuk informasi lebih lanjut tentang masalah deputi yang membingungkan, lihatCross-service bingung wakil pencegahan.

Gunakan konsol IAM untuk membuat dan mengonfigurasi peran dan izin. Untuk langkah-langkah yang membuat contoh peran yang me akses ke tabel Amazon DynamoDB, lihat the section called “Mengonfigurasi peran grup”. Untuk langkah umum, lihat Membuat peran untuk AWS layanan (konsol) dalam Panduan Pengguna IAM.

Setelah peran dikonfigurasi, gunakan konsol AWS IoT tersebut untuk menambahkan peran ke grup.

> **Note**
>
> Prosedur ini diperlukan hanya untuk memilih peran untuk grup. Tidak diperlukan setelah mengubah izin peran grup yang dipilih saat ini.

1. DiAWS IoTpanel navigasi konsol, di bawahKelola, PerluasPerangkat Greengrass, dan kemudian pilihGrup (V1).
2. Pilih grup target.
3. Pada halaman konfigurasi grup, pilihMelihat pengaturan.
4. Di bawahPeran grup, choose to add or change an:
   - Untuk menambahkan peran, pilihPeran asosiasidan kemudian pilih peran Anda dari daftar peran Anda. Ini adalah peran dalam Akun AWS Anda yang menentukan AWS IoT Greengrass sebagai entitas terpercaya.
   - Untuk memilih peran yang berbeda, pilihEdit perandan kemudian pilih peran Anda dari daftar peran Anda.
5. Pilih Save (Simpan).

Hapus peran grup Greengrass (konsol)

Ikuti langkah-langkah ini untuk melepaskan peran dari grup Greengrass.

1. DiAWS IoTpanel navigasi konsol, di bawahKelola, PerluasPerangkat Greengrass, dan kemudian pilihGrup (V1).
2. Pilih grup target.
3. Pada halaman konfigurasi grup, pilihMelihat pengaturan.
4. Di bawahPeran grup, choosePutus peran.
5. Di kotak dialog konfirmasi, pilihPutus peran. Langkah ini menghapus peran dari grup tetapi tidak menghapus peran. Jika ingin menghapus peran, gunakan konsol IAM.

Mengelola peran grup Greengrass (CLI)

Anda dapat menggunakan AWS CLI untuk tugas manajemen peran berikut:

- Dapatkan peran grup Greengrass Anda
- Buat peran grup Greengrass
- Hapus peran grup Greengrass

Dapatkan peran grup Greengrass (CLI)

Ikuti langkah-langkah ini untuk mengetahui apakah grup Greengrass memiliki peran yang terkait.

1. Dapatkan ID grup target dari daftar grup Anda.

   ```
   aws greengrass list-groups
   ```

   Berikut ini adalah contoh `list-groups` respons. Setiap grup dalam respon mencakup properti `Id` yang berisi ID grup.

   ```
   {
       "Groups": [
           {
               "LatestVersionArn": "arn:aws:us-west-2:123456789012:/greengrass/
   groups/00dedaaa-ac16-484d-ad77-c3eedEXAMPLE/versions/4cbc3f07-fc5e-48c4-
   a50e-7d356EXAMPLE",
               "Name": "MyFirstGroup",
               "LastUpdatedTimestamp": "2019-11-11T05:47:31.435Z",
               "LatestVersion": "4cbc3f07-fc5e-48c4-a50e-7d356EXAMPLE",
               "CreationTimestamp": "2019-11-11T05:47:31.435Z",
               "Id": "00dedaaa-ac16-484d-ad77-c3eedEXAMPLE",
               "Arn": "arn:aws:us-west-2:123456789012:/greengrass/groups/00dedaaa-
   ac16-484d-ad77-c3eedEXAMPLE"
           },
           {
               "LatestVersionArn": "arn:aws:us-west-2:123456789012:/greengrass/
   groups/036ceaf9-9319-4716-ba2a-237f9EXAMPLE/versions/8fe9e8ec-64d1-4647-
   b0b0-01dc8EXAMPLE",
               "Name": "GreenhouseSensors",
               "LastUpdatedTimestamp": "2020-01-07T19:58:36.774Z",
   ```

```
            "LatestVersion": "8fe9e8ec-64d1-4647-b0b0-01dc8EXAMPLE",
            "CreationTimestamp": "2020-01-07T19:58:36.774Z",
            "Id": "036ceaf9-9319-4716-ba2a-237f9EXAMPLE",
            "Arn": "arn:aws:us-west-2:123456789012:/greengrass/
groups/036ceaf9-9319-4716-ba2a-237f9EXAMPLE"
        },
        ...
    ]
}
```

Untuk informasi lebih lanjut, termasuk contoh yang menggunakan opsi `query` untuk memfilter hasil, lihat the section called “Mendapatkan ID grup”.

2. Salin `Id` dari grup target dari output.
3. Dapatkan peran grup. Ganti *`grup-id`* dengan ID grup target.

```
aws greengrass get-associated-role --group-id group-id
```

Jika peran dikaitkan dengan grup Greengrass Anda, peran metadata berikut dikembalikan.

```
{
  "AssociatedAt": "timestamp",
  "RoleArn": "arn:aws:iam::account-id:role/path/role-name"
}
```

Jika grup Anda tidak memiliki peran terkait, kesalahan berikut akan dikembalikan.

```
An error occurred (404) when calling the GetAssociatedRole operation: You need to
 attach an IAM role to this deployment group.
```

Buat peran grup Greengrass (CLI)

Ikuti langkah-langkah ini untuk membuat peran dan mengaitkannya dengan grup Greengrass.

Untuk membuat peran grup menggunakan IAM

1. Buat peran dengan kebijakan kepercayaan yang mengizinkan AWS IoT Greengrass untuk mengambil peran. Contoh ini menciptakan peran bernama `MyGreengrassGroupRole`,

tetapi Anda dapat menggunakan nama yang berbeda. Kami menyarankan Anda juga memasukkan`aws:SourceArn`dan`aws:SourceAccount`kunci konteks kondisi global dalam kebijakan kepercayaan Anda untuk membantu mencegahdeputyMasalah keamanan. Kunci konteks kondisi membatasi akses untuk mengizinkan hanya permintaan yang berasal dari akun tertentu dan ruang kerja Greengrass. Untuk informasi lebih lanjut tentang masalah deputi yang membingungkan, lihatCross-service bingung wakil pencegahan.

Linux, macOS, or Unix

```
aws iam create-role --role-name MyGreengrassGroupRole --assume-role-policy-
document '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "greengrass.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "account-id"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:greengrass:region:account-id:/greengrass/
groups/group-id"
        }
      }
    }
  ]
}'
```

Windows command prompt

```
aws iam create-role --role-name MyGreengrassGroupRole --assume-role-
policy-document "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect
\":\"Allow\",\"Principal\":{\"Service\":\"greengrass.amazonaws.com\"},
\"Action\":\"sts:AssumeRole\",\"Condition\":{\"ArnLike\":{\"aws:SourceArn
\":\"arn:aws:greengrass:region:account-id:/greengrass/groups/group-id\"},
\"StringEquals\":{\"aws:SourceAccount\":\"account-id\"}}}]}"
```

2. Salin peran ARN dari peran metadata dalam output. Anda menggunakan ARN untuk mengasosiasikan peran dengan grup Anda.
3. Lampirkan kebijakan terkelola atau inline ke peran untuk mendukung kasus bisnis Anda. Sebagai contoh, jika fungsi Lambda yang ditetapkan pengguna membaca dari Amazon S3, Anda mungkin melampirkan `AmazonS3ReadOnlyAccess` kebijakan terkelola untuk peran.

```
aws iam attach-role-policy --role-name MyGreengrassGroupRole --policy-arn
 arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess
```

Jika berhasil, tidak ada respons yang dikembalikan.

Untuk mengasosiasikan peran dengan grup Greengrass Anda

1. Dapatkan ID grup target dari daftar grup Anda.

```
aws greengrass list-groups
```

Berikut ini adalah contoh `list-groups` respons. Setiap grup dalam respon mencakup properti `Id` yang berisi ID grup.

```
{
    "Groups": [
        {
            "LatestVersionArn": "arn:aws:us-west-2:123456789012:/greengrass/
groups/00dedaaa-ac16-484d-ad77-c3eedEXAMPLE/versions/4cbc3f07-fc5e-48c4-
a50e-7d356EXAMPLE",
            "Name": "MyFirstGroup",
            "LastUpdatedTimestamp": "2019-11-11T05:47:31.435Z",
            "LatestVersion": "4cbc3f07-fc5e-48c4-a50e-7d356EXAMPLE",
            "CreationTimestamp": "2019-11-11T05:47:31.435Z",
            "Id": "00dedaaa-ac16-484d-ad77-c3eedEXAMPLE",
            "Arn": "arn:aws:us-west-2:123456789012:/greengrass/groups/00dedaaa-
ac16-484d-ad77-c3eedEXAMPLE"
        },
        {
            "LatestVersionArn": "arn:aws:us-west-2:123456789012:/greengrass/
groups/036ceaf9-9319-4716-ba2a-237f9EXAMPLE/versions/8fe9e8ec-64d1-4647-
b0b0-01dc8EXAMPLE",
```

```
            "Name": "GreenhouseSensors",
            "LastUpdatedTimestamp": "2020-01-07T19:58:36.774Z",
            "LatestVersion": "8fe9e8ec-64d1-4647-b0b0-01dc8EXAMPLE",
            "CreationTimestamp": "2020-01-07T19:58:36.774Z",
            "Id": "036ceaf9-9319-4716-ba2a-237f9EXAMPLE",
            "Arn": "arn:aws:us-west-2:123456789012:/greengrass/
groups/036ceaf9-9319-4716-ba2a-237f9EXAMPLE"
        },
        ...
    ]
}
```

Untuk informasi lebih lanjut, termasuk contoh yang menggunakan opsi `query` untuk memfilter hasil, lihat the section called “Mendapatkan ID grup”.

2. Salin `Id` dari grup target dari output.
3. Associate peran dengan grup Anda. Ganti *`grup-id`* dengan ID dari grup target dan *`role-arn`* dengan ARN peran grup.

```
aws greengrass associate-role-to-group --group-id group-id --role-arn role-arn
```

Jika berhasil, respons berikut dikembalikan.

```
{
  "AssociatedAt": "timestamp"
}
```

Hapus peran grup Greengrass (CLI)

Ikuti langkah-langkah ini untuk memisahkan peran grup dari grup Greengrass Anda.

1. Dapatkan ID grup target dari daftar grup Anda.

```
aws greengrass list-groups
```

Berikut ini adalah contoh `list-groups` respons. Setiap grup dalam respon mencakup properti `Id` yang berisi ID grup.

```
{
    "Groups": [
        {
            "LatestVersionArn": "arn:aws:us-west-2:123456789012:/greengrass/
groups/00dedaaa-ac16-484d-ad77-c3eedEXAMPLE/versions/4cbc3f07-fc5e-48c4-
a50e-7d356EXAMPLE",
            "Name": "MyFirstGroup",
            "LastUpdatedTimestamp": "2019-11-11T05:47:31.435Z",
            "LatestVersion": "4cbc3f07-fc5e-48c4-a50e-7d356EXAMPLE",
            "CreationTimestamp": "2019-11-11T05:47:31.435Z",
            "Id": "00dedaaa-ac16-484d-ad77-c3eedEXAMPLE",
            "Arn": "arn:aws:us-west-2:123456789012:/greengrass/groups/00dedaaa-
ac16-484d-ad77-c3eedEXAMPLE"
        },
        {
            "LatestVersionArn": "arn:aws:us-west-2:123456789012:/greengrass/
groups/036ceaf9-9319-4716-ba2a-237f9EXAMPLE/versions/8fe9e8ec-64d1-4647-
b0b0-01dc8EXAMPLE",
            "Name": "GreenhouseSensors",
            "LastUpdatedTimestamp": "2020-01-07T19:58:36.774Z",
            "LatestVersion": "8fe9e8ec-64d1-4647-b0b0-01dc8EXAMPLE",
            "CreationTimestamp": "2020-01-07T19:58:36.774Z",
            "Id": "036ceaf9-9319-4716-ba2a-237f9EXAMPLE",
            "Arn": "arn:aws:us-west-2:123456789012:/greengrass/
groups/036ceaf9-9319-4716-ba2a-237f9EXAMPLE"
        },
        ...
    ]
}
```

Untuk informasi lebih lanjut, termasuk contoh yang menggunakan opsi `query` untuk memfilter hasil, lihat the section called “Mendapatkan ID grup”.

2. Salin `Id` dari grup target dari output.
3. Lepaskan peran dari grup Anda. Ganti *`grup-id`* dengan ID grup target.

```
aws greengrass disassociate-role-from-group --group-id group-id
```

Jika berhasil, respons berikut dikembalikan.

```
{
```

```
  "DisassociatedAt": "timestamp"
}
```

> **Note**
>
> Anda dapat menghapus peran grup jika tidak menggunakannya. Pertama gunakan delete-role-policy untuk melepaskan setiap kebijakan terkelola dari peran, lalu gunakan delete-role untuk menghapus peran. Untuk informasi lebih lanjut, lihat Menghapus peran atau profil instans dalam Panduan Pengguna IAM.

Lihat juga

- Topik yang terkait di Panduan Pengguna IAM
  - Membuat peran untuk mendelegasikan izin kepadaAWSlayanan
  - Mengubah peran
  - Menambahkan dan menghapus izin identitas IAM
  - Menghapus profil peran atau instans
- AWS IoT Greengrass perintah dalam AWS CLI Referensi Perintah
  - daftar-grup
  - associate-role-to-group
  - disassociate-role-from-group
  - get-associated-role
- Perintah IAM di Referensi PerintahAWS CLI
  - attach-role-policy
  - create-role
  - delete-role
  - delete-role-policy

Cross-service bingung wakil pencegahan

Masalah deputi yang bingung adalah masalah keamanan di mana entitas yang tidak memiliki izin untuk melakukan tindakan dapat memaksa entitas yang lebih istimewa untuk melakukan tindakan tersebut. MasukAWS, peniruan lintas layanan dapat mengakibatkan masalah wakil bingung. Peniruan

lintas layanan dapat terjadi ketika satu layanan (layanan panggilan) panggilan layanan lain (disebut layanan). Layanan panggilan dapat dimanipulasi untuk menggunakan izin untuk bertindak atas sumber daya pelanggan lain dengan cara yang seharusnya tidak memiliki izin untuk mengakses. Untuk mencegah hal ini,AWSmenyediakan alat yang membantu Anda melindungi data Anda untuk semua layanan dengan prinsipal layanan yang telah diberikan akses ke sumber daya di akun Anda.

Sebaiknya gunakan`aws:SourceArn`dan`aws:SourceAccount`kunci konteks kondisi global dalam kebijakan sumber daya untuk membatasi izin yangAWS IoT Greengrassmemberikan layanan lain untuk sumber daya. Jika Anda menggunakan kedua kunci konteks kondisi global,`aws:SourceAccount`nilai dan akun di`aws:SourceArn`nilai harus menggunakan ID akun yang sama bila digunakan dalam pernyataan kebijakan yang sama.

Nilai dari`aws:SourceArn`harus menjadi sumber daya pelanggan Greengrass yang terkait dengan`sts:AssumeRole`permintaan.

Cara paling efektif untuk melindungi dari masalah wakil bingung adalah dengan menggunakan`aws:SourceArn`kunci konteks kondisi global dengan ARN penuh sumber daya. Jika Anda tidak mengetahui ARN penuh dari sumber daya atau jika Anda menentukan beberapa sumber daya, gunakan`aws:SourceArn`kunci kondisi konteks global dengan wildcard (`*`) untuk bagian yang tidak diketahui dari ARN. Sebagai contoh, `arn:aws:greengrass:`*`region`*`:`*`account-id`*`:*`.

Untuk contoh kebijakan yang menggunakan`aws:SourceArn`dan`aws:SourceAccount`kunci konteks kondisi global, lihat topik berikut:

- Buat peran layanan Greengrass
- Buat peran grup Greengrass
- Membuat dan mengonfigurasi peran eksekusi IAM untuk deployment massal

Contoh kebijakan berbasis identitas untuk AWS IoT Greengrass

Secara default, pengguna dan IAM role tidak memiliki izin untuk membuat atau memodifikasi AWS IoT Greengrass sumber daya. Mereka juga tidak dapat melakukan tugas dengan menggunakan API AWS Management Console, AWS CLI, atau AWS. Administrator IAM harus membuat kebijakan IAM yang memberikan izin kepada pengguna dan peran untuk melakukan operasi API tertentu pada sumber daya yang diperlukan. Administrator kemudian harus melampirkan kebijakan tersebut ke pengguna IAM atau grup yang memerlukan izin tersebut.

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapusAWS IoT Greengrass sumber daya di akun Anda. Tindakan ini membuat Akun AWS Anda terkena biaya. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Memulai kebijakanAWS terkelola dan beralih ke izin paling sedikit hak istimewa — Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakanAWS terkelola yang memberikan izin untuk banyak kasus penggunaan umum. Mereka tersedia di AndaAkun AWS. Kami menyarankan Anda mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelolaAWS pelanggan yang spesifik untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat kebijakanAWSAWS terkelola atau kebijakan terkelola untuk fungsi pekerjaan di Panduan Pengguna IAM.
- Berikan izin akses kecil — Saat Anda menetapkan izin dengan kebijakan IAM, hanya berikan izin yang diperlukan untuk melaksanakan tugas. Anda melakukan ini dengan menentukan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, juga dikenal sebagai izin paling tidak memiliki hak istimewa. Untuk informasi selengkapnya tentang penggunaan IAM untuk menerapkan izin, lihat Kebijakan dan izin di IAM dalam Panduan Pengguna IAM.
- Gunakan ketentuan dalam kebijakan IAM untuk membatasi akses lebih lanjut — Anda dapat menambahkan kondisi pada kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Misalnya, Anda dapat menulis ketentuan kebijakan untuk menentukan bahwa semua permintaan harus dikirim menggunakan SSL. Anda juga dapat menggunakan kondisi untuk memberikan akses ke tindakan layanan jika digunakan melalui spesifikLayanan AWS, sepertiAWS CloudFormation. Untuk informasi lebih lanjut, lihat Elemen kebijakan IAM JSON: Syarat dalam Panduan Pengguna IAM.
- Gunakan IAM Access Analyzer untuk memvalidasi kebijakan IAM Anda untuk memastikan izin yang aman dan fungsional - IAM Access Analyzer memvalidasi kebijakan baru dan yang sudah ada sehingga kebijakan mematuhi bahasa kebijakan IAM (JSON) dan praktik terbaik IAM. IAM Access Analyzer menyediakan lebih dari 100 pemeriksaan kebijakan dan rekomendasi yang dapat ditindaklanjuti untuk membantu Anda membuat kebijakan yang aman dan fungsional. Untuk informasi selengkapnya, lihat validasi kebijakan IAM Access Analyzer di Panduan Pengguna IAM.
- Memerlukan otentikasi multi-faktor (MFA) — Jika Anda memiliki skenario yang mengharuskan pengguna IAM atau pengguna root di AndaAkun AWS, aktifkan MFA untuk keamanan tambahan. Untuk mewajibkan MFA saat operasi API dipanggil, tambahkan kondisi MFA ke kebijakan Anda.

Untuk informasi selengkapnya, lihat Mengonfigurasi akses API yang dilindungi MFA di Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik dalam IAM, lihat praktik terbaik keamanan di IAM dalam Panduan Pengguna IAM.

Kebijakan yang dikelola oleh AWS untuk AWS IoT Greengrass

AWS IoT Greengrass mempertahankan kebijakan terkelola AWS berikut yang dapat Anda gunakan untuk memberikan izin kepada pengguna IAM dan peran.

Kebijakan	Deskripsi
AWSGreengrassFullAccess	Mengizinkan semua tindakan AWS IoT Greengrass untuk semua sumber daya AWS Anda. Kebijakan ini direkomendasikan untuk AWS IoT Greengrass administrator layanan atau tujuan pengujian.
AWSGreengrassReadOnlyAccess	Mengizinkan `List` dan `Get` AWS IoT Greengrass tindakan untuk semua sumber daya AWS Anda.
AWSGreengrassResourceAccessRolePolicy	Mengizinkan akses ke sumber daya dari AWS layanan termasuk AWS Lambda dan AWS IoT Bayangan Perangkat. Ini adalah kebijakan default yang digunakan untuk Peran layanan Greengrass. Kebijakan ini dirancang untuk menyediakan kemudahan akses secara umum. Anda dapat menentukan kebijakan kustom yang lebih ketat.
GreenGrassotaUpdateArtifactAccess	Mengizinkan akses baca saja ke over-the-air (OTA) memperbarui artefak untuk perangkat lunakAWS IoT Greengrass Core di semuaWila yah AWS s.

Contoh kebijakan

Contoh kebijakan yang ditetapkan pelanggan berikut memberikan izin untuk skenario umum.

Contoh

- Izinkan para pengguna untuk melihat izin mereka sendiri

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM menggunakan contoh dokumen kebijakan JSON ini, lihat Membuat kebijakan pada tab JSON dalam Panduan Pengguna IAM.

Izinkan para pengguna untuk melihat izin mereka sendiri

Contoh ini menunjukkan cara Anda dapat membuat kebijakan yang mengizinkan para pengguna IAM untuk melihat kebijakan inline dan terkelola yang dilampirkan ke identitas pengguna mereka. Kebijakan ini mencakup izin untuk menyelesaikan tindakan pada konsol atau secara terprogram menggunakan API AWS CLI atau AWS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
```

```
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Pemecahan masalah identitas dan akses untuk AWS IoT Greengrass

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang Anda mungkin temukan ketika bekerja dengan AWS IoT Greengrass dan IAM.

Masalah

- Saya tidak diotorisasi untuk melakukan tindakan di AWS IoT Greengrass
- Kesalahan: Greengrass tidak berwenang untuk menganggap peran layanan yang terkait dengan akun ini atau error: Greengrass tidak berwenang untuk menganggap peran layanan Gagal: Peran layanan TES tidak terkait dengan akun ini.
- Kesalahan: Izin ditolak ketika mencoba menggunakan peran arn:iam: :role/ <account-id>untuk <role-name>akses s3 url https://-greengrass-updates.s3<region>. <region>.amazonaws.com/inti// <architecture>greengrass-core- <distribution-version>.tar.gz.
- Bayangan perangkat tidak sinkron dengan cloud.
- Saya tidak berwenang untuk melakukan iam:PassRole
- Saya seorang administrator dan ingin mengizinkan orang lain mengakses AWS IoT Greengrass
- Saya ingin mengizinkan orang di luar Akun AWS saya untuk mengakses sumber daya AWS IoT Greengrass saya

Untuk bantuan pemecahan masalah umum, lihat Memecahkan masalah.

Saya tidak diotorisasi untuk melakukan tindakan di AWS IoT Greengrass

Jika Anda menerima kesalahan yang menyatakan bahwa Anda tidak terotorisasi untuk melakukan tindakan, maka Anda harus menghubungi administrator untuk mendapatkan bantuan. Administrator Anda adalah orang yang memberikan nama pengguna dan kata sandi Anda.

Contoh kesalahan berikut terjadi ketika `mateojackson` pengguna IAM mencoba melihat detail tentang versi definisi fungsi, tetapi tidak memiliki `greengrass:GetCoreDefinitionVersion` izin.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 greengrass:GetCoreDefinitionVersion on resource: resource: arn:aws:greengrass:us-
west-2:123456789012:/greengrass/definition/cores/78cd17f3-bc68-ee18-47bd-5bda5EXAMPLE/
versions/368e9ffa-4939-6c75-859c-0bd4cEXAMPLE
```

Dalam hal ini, Mateo meminta administratornya untuk memperbarui kebijakannya untuk mengizinkannya mengakses `arn:aws:greengrass:us-west-2:123456789012:/greengrass/definition/cores/78cd17f3-bc68-ee18-47bd-5bda5EXAMPLE/versions/368e9ffa-4939-6c75-859c-0bd4cEXAMPLE` sumber daya menggunakan `greengrass:GetCoreDefinitionVersion` tindakan.

Kesalahan: Greengrass tidak berwenang untuk menganggap peran layanan yang terkait dengan akun ini atau error: Greengrass tidak berwenang untuk menganggap peran layanan Gagal: Peran layanan TES tidak terkait dengan akun ini.

Solusi: Anda mungkin melihat kesalahan ini ketika deployment gagal. Periksa apakah peran layanan Greengrass dikaitkan dengan Akun AWS Anda saat ini Wilayah AWS. Untuk informasi lebih lanjut, lihat the section called “Mengelola peran layanan (CLI)” atau the section called “Mengelola peran layanan (konsol)”.

Kesalahan: Izin ditolak ketika mencoba menggunakan peran arn:iam: :role/ <account-id>untuk <role-name>akses s3 url https://-greengrass-updates.s3<region>. <region>.amazonaws.com/inti//<architecture>greengrass-core- <distribution-version>.tar.gz.

Solusi: Anda mungkin melihat kesalahan ini ketika over-the-air Pembaruan (OTA) gagal. Dalam kebijakan peran signer, tambahkan target Wilayah AWS sebagai `Resource`. Peran signer ini digunakan untuk presign S3 URL untuk AWS IoT Greengrass memperbarui perangkat lunak. Untuk informasi lebih lanjut, lihat Peran S3 URL signer.

Bayangan perangkat tidak sinkron dengan cloud.

Solusi: Pastikan bahwaAWS IoT Greengrassmemiliki izin untuk`iot:UpdateThingShadow`dan`iot:GetThingShadow`tindakan diPeran layanan Greengrass. Jika peran layanan menggunakan `AWSGreengrassResourceAccessRolePolicy` kebijakan terkelola, izin ini disertakan secara default.

Lihat Memecahkan masalah timeout sinkronisasi bayangan.

Berikut adalah masalah IAM umum yang Anda munkin hadapi ketika bekerja dengan AWS IoT Greengrass.

Saya tidak berwenang untuk melakukan iam:PassRole

Jika Anda menerima kesalahan bahwa Anda tidak terotorisasi untuk melakukan kesalahan bahwa Anda tidak terotorisasi untuk melakukan`iam:PassRole`tindakan, kebijakan Anda harus diperbarui untuk mengizinkan Anda dapat meneruskan peranAWS IoT Greengrass.

BeberapaLayanan AWSAnda dapat meneruskan peran yang sudah ada ke layanan tersebut alih-alih membuat peran layanan atau peran tertaut layanan baru atau peran tertaut-layanan. Untuk melakukannya, Anda harus memiliki izin untuk meneruskan peran ke layanan.

Contoh kesalahan berikut terjadi ketika pengguna IAM bernama `marymajor` mencoba menggunakan konsol tersebut untuk melakukan tindakan di AWS IoT Greengrass. Namun, tindakan tersebut mengharuskan layanan untuk memiliki izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut ke layanan.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dalam kasus ini, kebijakan Mary harus diperbarui untuk mengizinkannya untuk melakukan`iam:PassRole`tindakan.

Jika Anda membutuhkan bantuan, hubungiAWSadministrator. Administrator Anda adalah orang yang memberikan kredensyal masuk.

Saya seorang administrator dan ingin mengizinkan orang lain mengakses AWS IoT Greengrass

Untuk mengizinkan orang lain mengakses AWS IoT Greengrass, Anda harus membuat entitas IAM (pengguna atau peran) untuk orang atau aplikasi yang memerlukan akses. Mereka akan menggunakan kredensial untuk entitas tersebut untuk mengakses AWS. Anda kemudian harus melampirkan kebijakan yang memberi mereka izin yang tepat di AWS IoT Greengrass.

Untuk segera memulai, lihat Membuat pengguna dan grup IAM pertama Anda yang didelegasikan di Panduan Pengguna IAM.

Saya ingin mengizinkan orang di luar Akun AWS saya untuk mengakses sumber daya AWS IoT Greengrass saya

Anda dapat membuat IAM role yang dapat digunakan pengguna di akun lain atau orang di luar organisasi Anda untuk mengakses sumber daya AWS Anda. Anda dapat menentukan siapa yang dipercaya untuk menjalankan peran tersebut. Untuk informasi lebih lanjut, lihat Menyediakan akses ke pengguna IAM di pengguna lain Akun AWS yang kamu miliki dan Menyediakan akses ke akun Amazon Web Services yang dimiliki oleh pihak ketiga di Panduan Pengguna IAM.

AWS IoT Greengrass tidak mendukung akses lintas akun berdasarkan kebijakan berbasis sumber daya atau daftar kontrol akses (ACL).

Validasi kepatuhan untuk AWS IoT Greengrass

Untuk mempelajari apakah Layanan AWS berada dalam lingkup program kepatuhan khusus, lihat Layanan AWS di Scope oleh Program Program Kepatuhan yang Anda minati. Untuk informasi umum, lihat Program Kepatuhan AWS .

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Mengunduh Laporan di AWS Artifact.

Tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS ditentukan berdasarkan sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, serta hukum dan peraturan yang berlaku. AWS menyediakan sumber daya berikut untuk membantu kepatuhan:

- Panduan Mulai Cepat Keamanan dan Kepatuhan – Panduan deployment ini membahas pertimbangan arsitektur dan menyediakan langkah-langkah untuk melakukan deployment lingkungan dasar di AWS yang menjadi fokus keamanan dan kepatuhan.
- Merancang Keamanan dan Kepatuhan HIPAA di Amazon Web Services – Laporan resmi ini menjelaskan cara perusahaan dapat menggunakan AWS untuk membuat aplikasi yang memenuhi syarat HIPAA.

  > **Note**
  >
  > Tidak semua Layanan AWS memenuhi syarat HIPAA. Untuk informasi selengkapnya, lihat Referensi Layanan yang Memenuhi Syarat HIPAA.

- Sumber Daya Kepatuhan AWS – Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.

- Panduan Kepatuhan Pelanggan AWS – Pahami model tanggung jawab bersama melalui lensa kepatuhan. Panduan ini merangkum praktik terbaik untuk mengamankan Layanan AWS dan memetakan panduan kontrol keamanan di banyak kerangka kerja (termasuk National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Council (PCI), dan International Organization for Standardization (ISO)).
- Mengevaluasi Sumber Daya dengan Aturan dalam Panduan Developer AWS Config – Layanan AWS Config menilai seberapa baik konfigurasi sumber daya Anda dalam mematuhi praktik-praktik internal, pedoman industri, dan regulasi internal.
- AWS Security Hub – Layanan AWS ini memberikan pandangan komprehensif tentang status keamanan Anda di dalam AWS. Security Hub menggunakan kontrol keamanan untuk mengevaluasi sumber daya AWS Anda dan memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik. Untuk daftar layanan dan kontrol yang didukung, lihat Referensi kontrol Security Hub.
- AWS Audit Manager – Layanan AWS ini akan membantu Anda untuk terus-menerus mengaudit penggunaan AWS untuk menyederhanakan cara Anda mengelola risiko dan kepatuhan terhadap regulasi dan standar industri.

Ketahanan di AWS IoT Greengrass

Infrastruktur global AWS dibangun di sekitar Wilayah Amazon Web Services dan Availability Zone. Setiap Wilayah AWS menyediakan banyak Availability Zone yang terpisah dan terisolasi secara fisik, yang terhubung dengan jaringan yang memiliki latensi rendah, throughput tinggi, dan sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang serta mengoperasikan aplikasi dan basis data yang secara otomatis melakukan fail over di antara zona tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur pusat data tunggal atau multi tradisional.

Untuk informasi lebih lanjut tentang Amazon Web Services Regions and Availability Zones, lihat AWS Infrastrukur Global.

Selain infrastruktur global AWS tersebut, AWS IoT Greengrass menawarkan beberapa fitur untuk membantu mendukung ketahanan data dan kebutuhan pencadangan Anda.

- Jika inti kehilangan konektivitas internet, perangkat klien dapat terus berkomunikasi melalui jaringan lokal.
- Anda dapat mengonfigurasi core untuk menyimpan pesan yang belum diproses yang ditujukan untuk AWS Cloud target dalam cache penyimpanan lokal sebagai ganti penyimpanan dalam

memori. Cache penyimpanan lokal dapat bertahan selama core me-restart (sebagai contoh, setelah deployment grup atau perangkat reboot), sehingga AWS IoT Greengrass dapat terus memproses pesan yang ditujukan untuk AWS IoT Core. Untuk informasi selengkapnya, lihat the section called “Antrean pesan MQTT”.

- Anda dapat mengonfigurasi core untuk membuat sesi persisten dengan AWS IoT Core broker pesan. Ini mengizinkan core untuk menerima pesan yang dikirim saat core sedang offline. Untuk informasi selengkapnya, lihat the section called “Sesi persisten MQTT dengan AWS IoT Core”.
- Anda dapat mengonfigurasi grup Greengrass untuk menulis catatan ke sistem file lokal dan CloudWatch Log. Jika inti kehilangan konektivitas, logging lokal dapat berlanjut,tetapi CloudWatch log dikirim dengan jumlah pengulangan terbatas. Setelah retries habis, acara dihentikan. Anda harus juga menyadari batasan pencatatan.
- Anda dapat menulis fungsi Lambda yang membaca streaming Pengelola pengaliran dan mengirim data ke tujuan penyimpanan lokal.

Keamanan infrastruktur dalam AWS IoT Greengrass

Sebagai layanan terkelola,AWS IoT Greengrassdilindungi olehAWSkeamanan jaringan global. Untuk informasi tentangAWSlayanan keamanan dan bagaimanaAWSmelindungi infrastruktur, lihatAWSKeamanan Cloud. Untuk mendesainAWSlingkungan menggunakan praktik terbaik untuk keamanan infrastruktur, lihatPerlindungan InfrastrukturdiPilar KeamananAWSKerangka Kerja yang Diarsiteksikan dengan Baik.

Anda menggunakan panggilan API AWS yang dipublikasikan untuk mengakses AWS IoT Greengrass melalui jaringan. Klien harus mendukung hal berikut:

- Transport Layer Security (TLS). Kami membutuhkan TLS 1.2 dan merekomendasikan TLS 1.3.
- Suite cipher dengan kerahasiaan maju sempurna (PFS) seperti DHE (Ephemeral Diffie-Hellman) atau ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Sebagian besar sistem modern seperti Java 7 dan sistem yang lebih baru mendukung mode ini.

Selain itu, permintaan harus ditandatangani menggunakan access key ID dan secret access key yang terkait dengan principal IAM. Atau Anda bisa menggunakan AWS Security Token Service (AWS STS) untuk membuat kredensial keamanan sementara guna menandatangani permintaan.

Dalam lingkungan AWS IoT Greengrass, perangkat menggunakan sertifikat X.509 dan kunci kriptografi untuk terhubung dan diautentikasi ke AWS Cloud. Untuk informasi selengkapnya, lihat the section called “Autentikasi dan otorisasi perangkat”.

Analisis konfigurasi dan kerentanan dalam AWS IoT Greengrass

Lingkungan IoT dapat terdiri atas sejumlah besar perangkat yang memiliki beragam kemampuan, berumur panjang, dan didistribusikan secara geografis. Karakteristik ini membuat penyiapan perangkat menjadi kompleks dan rawan kesalahan. Dan karena perangkat sering dibatasi dalam daya komputasi, memori, dan kemampuan penyimpanan, ini membatasi penggunaan enkripsi dan bentuk keamanan lainnya pada perangkat itu sendiri. Selain itu, perangkat sering menggunakan perangkat lunak dengan kerentanan yang diketahui. Faktor-faktor ini membuat perangkat IoT menjadi target yang menarik bagi peretas dan membuatnya sulit untuk mengamankannya secara berkelanjutan.

AWS IoT Device Defender mengatasi tantangan ini dengan menyediakan alat untuk mengidentifikasi masalah keamanan dan penyimpangan dari praktik terbaik. Anda dapat menggunakan AWS IoT Device Defender untuk menganalisis, mengaudit, dan memantau perangkat yang terhubung untuk mendeteksi perilaku abnormal, dan mengurangi risiko keamanan. AWS IoT Device Defender dapat mengaudit perangkat untuk memastikannya mematuhi praktik terbaik keamanan dan mendeteksi perilaku abnormal pada perangkat. Hal ini mengizinkan untuk menerapkan kebijakan keamanan yang konsisten di seluruh perangkat Anda dan merespons dengan cepat ketika perangkat disusupi. Dalam hubungan dengan AWS IoT Core, AWS IoT Greengrass menghasilkan ID klien yang dapat diprediksi yang dapat Anda gunakan dengan fitur AWS IoT Device Defender ini. Untuk informasi lebih lanjut, lihat AWS IoT Device Defender dalam Panduan DeveloperAWS IoT Core.

Dalam AWS IoT Greengrass lingkungan, Anda harus mengetahui pertimbangan berikut:

- Merupakan tanggung jawab Anda untuk mengamankan perangkat fisik Anda, sistem file pada perangkat Anda, dan jaringan lokal.
- AWS IoT Greengrass tidak menegakkan isolasi jaringan untuk komponen fungsi Lambda yang ditetapkan pengguna, apakah mereka berjalan atau tidak dalam kontainer Greengrass. Oleh karena itu, mungkin bagi komponen fungsi Lambda untuk berkomunikasi dengan proses lain yang berjalan di sistem atau di luar melalui jaringan.

Jika Anda kehilangan kendali atas perangkat inti Greengrass dan Anda ingin mencegah perangkat klien mentransmisikan data ke inti, lakukan hal berikut:

1. Hapus core Greengrass dari grup Greengrass.
2. Putar sertifikat CA grup. Dalam konsol AWS IoT tersebut, Anda dapat memutar sertifikat CA pada halaman grup Pengaturan ini. Di AWS IoT Greengrass API, Anda dapat menggunakan CreateGroupCertificateAuthoritytindakan.

   Kami juga merekomendasikan untuk menggunakan enkripsi disk penuh jika hard drive perangkat core Anda lemah terhadap pencurian.

AWS IoT Greengrass dan titik akhir VPC antarmuka (AWS PrivateLink)

Anda dapat membuat koneksi privat antara VPC danAWS IoT Greengrasskontrol pesawat dengan membuatVPC endpoint antarmuka. Anda dapat menggunakan endpoint ini untuk mengelola grup, fungsi Lambda, penyebaran, dan sumber daya lainnya diAWS IoT Greengrasslayanan. Endpoint antarmuka didukung olehAWS PrivateLink, teknologi yang memungkinkan Anda untuk mengaksesAWS IoT GreengrassAPI secara privat tanpa Gateway internet, perangkat NAT, koneksi VPN, atauAWSKoneksi Direct Connect. Instans dalam VPC Anda tidak memerlukan alamat IP publik untuk berkomunikasi dengan API AWS IoT Greengrass. Lalu lintas antara VPC Anda dan AWS IoT Greengrass tidak meninggalkan jaringan Amazon.

> **Note**
>
> Saat ini, Anda tidak dapat mengkonfigurasi perangkat inti Greengrass untuk beroperasi sepenuhnya dalam VPC Anda.

Setiap titik akhir antarmuka diwakili oleh satu atau beberapa Antarmuka Jaringan Elastis di subnet Anda.

Untuk informasi selengkapnya, lihat Antarmuka VPC endpoint (AWS PrivateLink) dalam Panduan Pengguna Amazon VPC.

Topik

- Pertimbangan untuk VPC endpoint AWS IoT Greengrass
- Buat VPC endpoint antarmuka untukAWS IoT GreengrassOperasi bidang kendali
- Membuat kebijakan VPC endpoint untuk AWS IoT Greengrass

Pertimbangan untuk VPC endpoint AWS IoT Greengrass

Sebelum Anda mengatur VPC endpoint antarmuka untukAWS IoT Greengrass, tinjauanProperti endpoint antarmuka dan keterbatasandi dalamPanduan Pengguna Amazon VPC. Selain itu, perhatikan pertimbangan berikut ini:

- AWS IoT Greengrassmendukung panggilan ke semua tindakan API bidang kendali dari VPC Anda. Pesawat kontrol mencakup operasi sepertiCreateDeploymentdanStartBulkDeployment. Bidang kendalitidaktermasuk operasi sepertiGetDeploymentdanTemukan, yang merupakan operasi bidang data.
- VPC endpoint untukAWS IoT Greengrasssaat ini tidak didukungAWSWilayah Tiongkok.

Buat VPC endpoint antarmuka untukAWS IoT GreengrassOperasi bidang kendali

Anda dapat membuat VPC endpoint untukAWS IoT Greengrasscontrol plane menggunakan konsol Amazon VPC atauAWS Command Line Interface(AWS CLI). Untuk informasi lebih lanjut, lihat Membuat titik akhir antarmuka di Panduan Pengguna Amazon VPC.

Buat VPC endpoint untukAWS IoT Greengrassmenggunakan nama layanan berikut:

- com.amazonaws.*daerah*.greengrass

Jika Anda mengaktifkan DNS privat untuk titik akhir, Anda dapat membuat permintaan APIAWS IoT Greengrassmenggunakan nama DNS default untuk Wilayah, misalnya,`greengrass.us-east-1.amazonaws.com`. DNS privat diaktifkan secara default.

Untuk informasi lebih lanjut, lihat Mengakses layanan melalui titik akhir antarmuka dalam Panduan Pengguna Amazon VPC.

Membuat kebijakan VPC endpoint untuk AWS IoT Greengrass

Anda dapat melampirkan kebijakan titik akhir ke VPC endpoint yang mengontrol akses keAWS IoT Greengrassoperasi bidang kendali. Kebijakan menentukan informasi berikut ini:

- Prinsip yang dapat melakukan tindakan.
- Tindakan yang dapat dilakukan oleh prinsip.

- Sumber daya yang dapat digunakan untuk melakukan tindakan.

Untuk informasi selengkapnya, lihat Mengendalikan akses ke layanan dengan titik akhir VPC dalam Panduan Pengguna Amazon VPC.

Example Contoh: Kebijakan VPC endpoint untukAWS IoT Greengrasstindakan

Berikut adalah contoh kebijakan titik akhir untuk AWS IoT Greengrass. Jika dilampirkan ke sebuah titik akhir, kebijakan ini memberikan akses ke tindakan AWS IoT Greengrass yang terdaftar untuk semua yang utama di semua sumber daya.

```
{
    "Statement": [
        {
            "Principal": "*",
            "Effect": "Allow",
            "Action": [
                "greengrass:CreateDeployment",
                "greengrass:StartBulkDeployment"
            ],
            "Resource": "*"
        }
    ]
}
```

Praktik terbaik keamanan untuk AWS IoT Greengrass

Topik ini berisi praktik terbaik keamanan untuk AWS IoT Greengrass.

Berikan izin minimum yang memungkinkan

Ikuti prinsip hak istimewa paling sedikit dengan menggunakan seperangkat izin minimum pada IAM Role. Batasi penggunaan * wildcard untuk `Action` dan `Resource` di kebijakan IAM Anda. Sebaliknya, nyatakan serangkaian terbatas tindakan dan sumber daya bila memungkinkan. Untuk informasi lebih lanjut tentang hak istimewa minimum dan praktik terbaik kebijakan lainnya, lihat the section called “Praktik terbaik kebijakan”.

Praktik terbaik dengan hak istimewa paling sedikit juga berlaku untuk AWS IoT kebijakan yang Anda lampirkan ke perangkat inti dan klien Greengrass Anda.

Jangan me-hardcode kredensial dalam fungsi Lambda

Jangan me-hardcode kredensial dalam fungsi Lambda yang ditetapkan pengguna milik Anda. Untuk melindungi kredensial Anda dengan lebih baik:

- Untuk berinteraksi dengan AWS layanan, tentukan izin untuk tindakan tertentu dan sumber daya dalam Peran grup Greengrass.
- Gunakan Rahasia Lokal untuk menyimpan kredensial Anda. Atau, jika fungsi menggunakan AWS SDK, gunakan kredensial dari rantai penyedia kredensial default.

Jangan log informasi sensitif

Anda harus mencegah logging kredensial dan informasi pengenal pribadi (PII) lainnya. Kami menyarankan Anda menerapkan perlindungan berikut meskipun akses ke log lokal pada perangkat inti memerlukan hak akses root dan akses ke CloudWatch Log memerlukan izin IAM.

- Jangan gunakan informasi sensitif di jalur topik MQTT.
- Jangan gunakan informasi sensitif pada nama, jenis, dan atribut perangkat (hal) di registri AWS IoT Core ini.
- Jangan mencatat informasi sensitif dalam fungsi Lambda yang ditetapkan pengguna Anda.
- Jangan gunakan informasi sensitif dalam nama dan ID sumber daya Greengrass:
  - Konektor
  - Core
  - Perangkat
  - Fungsi
  - Grup
  - Pencatat
  - Sumber daya (lokal, machine learning, atau rahasia)
  - Langganan

Buat langganan yang ditargetkan

Langganan mengontrol aliran informasi dalam grup Greengrass dengan mendefinisikan cara pesan dipertukarkan antara layanan, perangkat, dan fungsi Lambda. Untuk memastikan bahwa aplikasi

hanya dapat melakukan apa yang dimaksudkan untuk dilakukan, langganan Anda harus mengizinkan penerbit untuk mengirim pesan ke topik tertentu saja, dan membatasi pelanggan untuk menerima pesan hanya dari topik yang diperlukan untuk fungsionalitas mereka.

Sinkronkan jam perangkat Anda

Penting untuk memiliki waktu yang akurat di perangkat Anda. Sertifikat X.509 memiliki tanggal dan waktu kedaluwarsa. Jam di perangkat Anda digunakan untuk memverifikasi bahwa sertifikat server masih valid. Jam perangkat dapat melayang dari waktu ke waktu atau baterai dapat habis.

Untuk informasi lebih lanjut, lihat praktik terbaik Sinkronkan jam perangkat di AWS IoT Core Panduan Developer.

Mengelola autentikasi perangkat dengan core Greengrass

Perangkat klien dapat menjalankan FreeRTOS atau menggunakan AWS IoTDevice SDK AWS IoT Greengrassatau Discovery API untuk mendapatkan informasi penemuan yang digunakan untuk menghubungkan dan mengautentikasi dengan inti dalam grup Greengrass yang sama. Informasi penemuan meliputi:

- Informasi konektivitas untuk inti Greengrass yang ada di grup Greengrass yang sama dengan perangkat klien. Informasi ini mencakup alamat host dan nomor port setiap titik akhir untuk perangkat core.
- Sertifikat CA grup yang digunakan untuk menandatangani sertifikat server MQTT lokal. Perangkat klien menggunakan sertifikat CA grup untuk memvalidasi sertifikat server MQTT yang disajikan oleh inti.

Berikut ini adalah praktik terbaik untuk perangkat klien untuk mengelola otentikasi timbal balik dengan inti Greengrass. Praktik ini dapat membantu mengurangi risiko Anda jika perangkat core Anda terganggu.

Validasi sertifikat server MQTT lokal untuk setiap koneksi.

> Perangkat klien harus memvalidasi sertifikat server MQTT yang disajikan oleh inti setiap kali mereka membuat koneksi dengan inti. Validasi ini adalah sisi perangkat klien dari otentikasi timbal balik antara perangkat inti dan perangkat klien. Perangkat klien harus dapat mendeteksi kegagalan dan mengakhiri koneksi.

Jangan hardcode informasi penemuan.

Perangkat klien harus mengandalkan operasi penemuan untuk mendapatkan informasi konektivitas inti dan sertifikat CA grup, bahkan jika inti menggunakan alamat IP statis. Perangkat klien tidak boleh melakukan hardcode informasi penemuan ini.

Perbarui informasi penemuan secara berkala.

Perangkat klien harus menjalankan penemuan secara berkala untuk memperbarui informasi konektivitas inti dan sertifikat CA grup. Kami menyarankan agar perangkat klien memperbarui informasi ini sebelum mereka membuat koneksi dengan inti. Karena durasi yang lebih pendek antara operasi penemuan dapat meminimalkan potensi waktu pemaparan Anda, kami menyarankan agar perangkat klien memutuskan sambungan dan menyambung kembali secara berkala untuk memicu pembaruan.

Jika Anda kehilangan kendali atas perangkat inti Greengrass dan Anda ingin mencegah perangkat klien mentransmisikan data ke inti, lakukan hal berikut:

1. Hapus core Greengrass dari grup Greengrass.
2. Putar sertifikat CA grup. Dalam konsol AWS IoT tersebut, Anda dapat memutar sertifikat CA pada halaman grup Pengaturan ini. Di AWS IoT Greengrass API, Anda dapat menggunakan CreateGroupCertificateAuthoritytindakan.

   Kami juga merekomendasikan untuk menggunakan enkripsi disk penuh jika hard drive perangkat core Anda lemah terhadap pencurian.

Untuk informasi selengkapnya, lihat the section called “Autentikasi dan otorisasi perangkat”.

Lihat juga

- Praktik terbaik keamanan di AWS IoT Core pada Panduan Developer AWS IoT
- Sepuluh aturan emas keamanan untuk solusi IoT Industri di Internet of Things di Blog Resmi AWS

Pencatatan dan pemantauan di AWS IoT Greengrass

Pemantauan adalah bagian penting dari pemeliharaan keandalan, ketersediaan, dan performa AWS IoT Greengrass serta solusi AWS Anda. Anda harus mengumpulkan data pemantauan dari semua bagian solusi AWS Anda agar dapat dengan lebih mudah melakukan debug jika terjadi kegagalan multititik. Namun sebelum Anda mulai memantau AWS IoT Greengrass, Anda harus membuat rencana pemantauan yang mencakup jawaban atas pertanyaan berikut:

- Apa tujuan pemantauan Anda?
- Sumber daya mana yang akan Anda pantau?
- Seberapa sering Anda akan memantau sumber daya ini?
- Alat pemantauan mana yang akan Anda gunakan?
- Siapa yang akan melakukan tugas pemantauan?
- Siapa yang harus diberi tahu saat terjadi kesalahan?

Alat pemantauan

AWS menyediakan berbagai alat yang dapat Anda gunakan untuk memantau AWS IoT Greengrass. Anda dapat mengonfigurasi beberapa alat ini agar melakukan pemantauan untuk Anda. Beberapa alat memerlukan intervensi manual. Kami menyarankan agar Anda mengautomasi tugas pemantauan sebanyak mungkin.

Anda dapat menggunakan alat pemantauan otomatis berikut untuk memantau AWS IoT Greengrass dan melaporkan masalah:

- Amazon CloudWatch Beberapa catatan— Memantau, menyimpan, dan mengakses file log Anda dariAWS CloudTrailatau sumber lainnya. Untuk informasi selengkapnya, lihatMemantau berkas logdiAmazon CloudWatch Panduan Pengguna.
- AWS CloudTrailPemantauan Log— Berbagi file log antara akun, memantau CloudTrail log file secara real time dengan mengirim mereka ke CloudWatch Log, menulis aplikasi pemrosesan log di Java, dan memvalidasi bahwa berkas log Anda tidak berubah setelah pengiriman oleh CloudTrail. Untuk informasi selengkapnya, lihatBekerja dengan CloudTrail berkas logdiAWS CloudTrailPanduan Pengguna.
- Amazon EventBridge— Gunakan EventBridge aturan acara agar mendapatkan pemberitahuan tentang perubahan status untuk deployment grup Greengrass atau panggilan API yang dicatat

dengan CloudTrail. Untuk informasi selengkapnya, lihatthe section called “Dapatkan notifikasi deployment”atauApa itu Amazon EventBridge?diAmazon EventBridge Panduan Pengguna.

- Telemetri kesehatan sistem Greengrass — Berlangganlah untuk menerima data telemetri yang dikirim dari inti Greengrass. Untuk informasi selengkapnya, lihat the section called “Mengumpulkan data telemetri kondisi sistem”.
- Pemeriksaan kondisi lokal – Gunakan API kondisi untuk mendapatkan snapshot dari status lokal proses AWS IoT Greengrass pada perangkat core. Untuk informasi selengkapnya, lihat the section called “Memanggil API pemeriksaan kondisi lokal”.

Lihat juga

- the section called “Pemantauan dengan AWS IoT Greengrass log”
- the section called “Mencatat log panggilan API AWS IoT Greengrass dengan AWS CloudTrail”
- the section called “Dapatkan notifikasi deployment”

Pemantauan dengan AWS IoT Greengrass log

AWS IoT Greengrass terdiri dari layanan cloud dan AWS IoT Greengrass perangkat lunak core. Perangkat lunak AWS IoT Greengrass Core dapat menulis log ke Amazon CloudWatch dan ke sistem file lokal perangkat inti Anda. Fungsi dan konektor Lambda yang berjalan pada inti juga dapat menulis CloudWatch log ke Log dan sistem file lokal. Anda dapat menggunakan log untuk memantau acara dan meyelesaikan masalah. Semua entri log AWS IoT Greengrass termasuk timestamp, tingkat log, dan informasi tentang acara. Perubahan pada pengaturan pencatatan berlaku setelah Anda men-deploy grup.

Pencatatan dikonfigurasi pada tingkat grup. Untuk langkah-langkah yang menunjukkan cara mengonfigurasi pencatatan untuk grup Greengrass, lihat the section called “Konfigurasi pencatatan untuk AWS IoT Greengrass”.

Mengakses Log CloudWatch

Jika Anda mengonfigurasi CloudWatch logging, Anda dapat melihat log di halaman Log di CloudWatch konsol Amazon. Grup log untuk log AWS IoT Greengrass menggunakan konvensi penamaan berikut:

```
/aws/greengrass/GreengrassSystem/greengrass-system-component-name
```

```
/aws/greengrass/Lambda/aws-region/account-id/lambda-function-name
```

Setiap grup log berisi aliran log yang menggunakan Konvensi penamaan berikut:

```
date/account-id/greengrass-group-id/name-of-core-that-generated-log
```

Pertimbangan berikut berlaku saat Anda menggunakan CloudWatch Log:

- Log dikirim ke CloudWatch Log dengan jumlah percobaan ulang terbatas jika tidak ada konektivitas internet. Setelah percobaan ulang habis, kegiatan ini dibatalkan.
- Transaksi, memori, dan keterbatasan lainnya berlaku. Untuk informasi selengkapnya, lihat the section called “Batasan pencatatan”.
- Peran grup Greengrass Anda harus AWS IoT Greengrass memungkinkan untuk menulis ke Log. CloudWatch Untuk memberikan izin, menanamkan kebijakan di barisan berikut dalam peran grup Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:*:*:*"
            ]
        }
    ]
}
```

> **Note**
>
> Anda dapat memberikan akses yang terperinci ke sumber daya log Anda. Untuk informasi selengkapnya, lihat Menggunakan kebijakan berbasis identitas (kebijakan IAM) untuk Log CloudWatch di Panduan Pengguna Amazon. CloudWatch

Peran grup adalah IAM role yang Anda buat dan lampirkan ke grup Greengrass Anda. Anda dapat menggunakan konsol atau API AWS IoT Greengrass untuk mengelola peran grup.

Menggunakan konsol

1. Di panel navigasi AWS IoT konsol, di bawah Kelola, perluas perangkat Greengrass, lalu pilih Grup (V1).
2. Pilih grup target.
3. Pilih Lihat pengaturan. Di bawah Peran grup, Anda dapat melihat, mengaitkan, atau memisahkan peran grup.

   Untuk langkah-langkah yang menunjukkan cara melampirkan peran grup, lihat peran grup.

Menggunakan CLI

- Untuk menemukan peran grup, gunakan get-associated-roleperintah.
- Untuk melampirkan peran grup, gunakan associate-role-to-groupperintah.
- Untuk menghapus peran grup, gunakan disassociate-role-from-groupperintah.

  Untuk mempelajari cara mendapatkan ID grup agar dapat digunakan dengan perintah ini, lihat the section called “Mendapatkan ID grup”.

Mengakses log sistem file

Jika Anda mengonfigurasi pencatatan sistem file, file log disimpan di bawah *greengrass-root*/ggc/var/log pada perangkat core. Berikut ini adalah struktur direktori tingkat tinggi:

```
greengrass-root/ggc/var/log
    - crash.log
    - system
        - log files for each Greengrass system component
    - user
        - region
            - account-id
                - log files generated by each user-defined Lambda function
            - aws
                - log files generated by each connector
```

> **Note**
>
> Secara default, *greengrass-root* adalah direktori `/greengrass` ini. Jika menulis direktori dikonfigurasi, maka log berada di bawah direktori itu.

Pertimbangan berikut berlaku saat Anda menggunakan log sistem file:

- Membaca log AWS IoT Greengrass pada sistem file memerlukan izin root.
- AWS IoT Greengrass mendukung rotasi berbasis ukuran dan pembersihan otomatis ketika jumlah data log mendekati batas yang dikonfigurasi.
- File `crash.log` tersedia hanya dalam log sistem file. Log ini tidak ditulis ke CloudWatch Log.
- Pembatasan penggunaan disk berlaku. Untuk informasi selengkapnya, lihat the section called "Batasan pencatatan".

> **Note**
>
> Catatan untuk perangkat lunak AWS IoT Greengrass core v1.0 disimpan di bawah direktori *greengrass-root*`/var/log` ini.

Konfigurasi pencatatan default

Jika pengaturan pencatatan tidak dikonfigurasi secara eksplisit, AWS IoT Greengrass akan menggunakan konfigurasi pencatatan default berikut setelah deployment grup pertama.

AWS IoT Greengrass Komponen Sistem

- Jenis - `FileSystem`
- Komponen - `GreengrassSystem`
- Tingkat - `INFO`
- Yang lebih besar - `128 KB`

Fungsi Lambda yang ditentukan pengguna

- Jenis - `FileSystem`
- Komponen - `Lambda`
- Tingkat - `INFO`
- Yang lebih besar - `128 KB`

> **Note**
>
> Sebelum deployment pertama, hanya komponen sistem yang menulis log ke sistem file karena tidak ada fungsi Lambda yang ditentukan pengguna di-deploy.

Konfigurasi pencatatan untuk AWS IoT Greengrass

Anda dapat menggunakan konsol AWS IoT tersebut atau AWS IoT Greengrass API untuk mengonfigurasi AWS IoT Greengrass pencatatan.

> **Note**
>
> Agar dapat menulis log ke CloudWatch Log, peran grup Anda harus mengizinkan tindakan CloudWatch Log yang diperlukan. AWS IoT Greengrass

Konfigurasi pencatatan (konsol)

Anda dapat mengonfigurasi pencatatan di grup halaman Pengaturan ini.

1. Di panel navigasi AWS IoT konsol, di bawah Kelola, perluas perangkat Greengrass, lalu pilih Grup (V1).

2. Pilih grup tempat Anda ingin mengonfigurasi pencatatan.
3. Pada halaman konfigurasi grup, pilih tab Log.
4. Pilih lokasi pencatatan, sebagai berikut:

   - Untuk mengonfigurasi CloudWatch logging, untuk konfigurasi CloudWatch log, pilih Edit.
   - Untuk mengonfigurasi pencatatan sistem file, untuk konfigurasi log Lokal, pilih Edit.

   Anda dapat mengonfigurasi pencatatan untuk satu atau kedua lokasi.
5. Dalam modal konfigurasi edit log, pilih tingkat log sistem Greengrass atau tingkat log fungsi Lambda Pengguna. Anda bisa memilih satu atau kedua komponen.
6. Pilih tingkat terendah acara yang ingin Anda masukkan log. acara di bawah ambang batas ini disaring dan tidak disimpan.
7. Pilih Simpan. Perubahan berlaku setelah Anda men-deploy grup.

Konfigurasi pencatatan (API)

Anda dapat menggunakan pencatat AWS IoT Greengrass API untuk mengonfigurasi pencatatan pemrograman. Misalnya, gunakan tindakan `CreateLoggerDefinition` untuk membuat definisi pencatat berdasarkan muatan `LoggerDefinitionVersion` ini, yang menggunakan sintaks berikut:

```
{
  "Loggers": [
    {
      "Id": "string",
      "Type": "FileSystem|AWSCloudWatch",
      "Component": "GreengrassSystem|Lambda",
      "Level": "DEBUG|INFO|WARN|ERROR|FATAL",
      "Space": "integer"
    },
    {
      "Id": "string",
      ...
    }
  ]
}
```

`LoggerDefinitionVersion` adalah array dari satu atau lebih objek `Logger` yang memiliki properti berikut:

`Id`

Pengidentifikasi untuk pencatat.

`Type`

Mekanisme penyimpanan untuk log acara. Saat `AWSCloudWatch` digunakan, peristiwa log dikirim ke CloudWatch Log. Saat `FileSystem` digunakan, log acara disimpan pada sistem file lokal.

Nilai yang valid: `AWSCloudWatch`, `FileSystem`

`Component`

Sumber log acara. Saat `GreengrassSystem` digunakan, acara dari komponen sistem Greengrass dicatat. Saat `Lambda` digunakan, acara dari fungsi Lambda yang ditentukan pengguna dicatat.

Nilai yang valid: `GreengrassSystem`, `Lambda`

`Level`

Ambang tingkat log. Log acara di bawah ambang batas ini disaring dan tidak disimpan.

Nilai yang valid: `DEBUG`, `INFO` (direkomendasikan), `WARN`, `ERROR`, `FATAL`

`Space`

Jumlah maksimum penyimpanan lokal, dalam KB, digunakan untuk menyimpan log. Bidang ini hanya berlaku bila `Type` diatur ke `FileSystem`.

Contoh konfigurasI

Contoh `LoggerDefinitionVersion` berikut menentukan konfigurasi pencatatan yang:

- Mengaktifkan sistem file `ERROR` dan pencatatan di atas untuk komponen sistem AWS IoT Greengrass ini.
- Mengaktifkan pencatatan sistem file `INFO` (dan di atas) untuk fungsi Lambda yang ditentukan pengguna.
- Mengaktifkan CloudWatch `INFO` (dan di atas) logging untuk fungsi Lambda yang ditentukan pengguna.

```
{
```

```
  "Name": "LoggingExample",
  "InitialVersion": {
    "Loggers": [
      {
        "Id": "1",
        "Component": "GreengrassSystem",
        "Level": "ERROR",
        "Space": 10240,
        "Type": "FileSystem"
      },
      {
        "Id": "2",
        "Component": "Lambda",
        "Level": "INFO",
        "Space": 10240,
        "Type": "FileSystem"
      },
      {
        "Id": "3",
        "Component": "Lambda",
        "Level": "INFO",
        "Type": "AWSCloudWatch"
      }
    ]
  }
}
```

Setelah Anda membuat versi definisi pencatat, Anda dapat menggunakan versi ARN untuk membuat versi grup sebelum men-deploy grup.

Batasan pencatatan

AWS IoT Greengrass memiliki batasan pencatatan berikut.

Transaksi per detik (TPS)

Saat login ke CloudWatch diaktifkan, komponen logging akan mengumpulkan peristiwa log secara lokal sebelum mengirimnya CloudWatch, sehingga Anda dapat masuk dengan kecepatan lebih tinggi dari lima permintaan per detik per aliran log.

Memori

Jika AWS IoT Greengrass dikonfigurasi untuk mengirim log ke CloudWatch dan fungsi Lambda mencatat lebih dari 5 MB/detik untuk jangka waktu yang lama, pipeline pemrosesan internal akhirnya terisi. Kasus terburuk teoritis adalah 6 MB per fungsi Lambda.

Clock skew

Saat login ke CloudWatch diaktifkan, komponen logging menandatangani permintaan untuk CloudWatch menggunakan proses penandatanganan Signature Version 4 normal. Jika waktu sistem pada perangkat AWS IoT Greengrass core tidak sinkron lebih dari 15 menit, maka permintaan ditolak.

Penggunaan disk

Gunakan rumus berikut untuk menghitung jumlah maksimum total penggunaan disk dalam pencatatan.

```
greengrass-system-component-space * 8    // 7 if automatic IP detection is disabled
  + 128KB                                // the internal log for the local logging
 component
  + lambda-space * lambda-count          // different versions of a Lambda function are
 treated as one
```

Di mana:

`greengrass-system-component-space`

Jumlah maksimum penyimpanan lokal untuk log komponen sistem AWS IoT Greengrass ini.

`lambda-space`

Jumlah maksimum penyimpanan lokal untuk log fungsi Lambda.

`lambda-count`

Jumlah fungsi Lambda yang di-deploy.

Kehilangan log

Jika perangkat AWS IoT Greengrass inti Anda dikonfigurasi untuk masuk hanya ke CloudWatch dan tidak ada konektivitas internet, Anda tidak memiliki cara untuk mengambil log yang saat ini ada di memori.

Ketika fungsi Lambda dihentikan (misalnya, selama penerapan), log selama beberapa detik tidak ditulis. CloudWatch

CloudTrail log

AWS IoT Greengrass berjalan dengan AWS CloudTrail, layanan yang menyediakan log tindakan yang diambil oleh pengguna, peran, atau layanan AWS di AWS IoT Greengrass. Lihat informasi yang lebih lengkap di the section called “Mencatat log panggilan API AWS IoT Greengrass dengan AWS CloudTrail”.

Mencatat log panggilan API AWS IoT Greengrass dengan AWS CloudTrail

AWS IoT Greengrassterintegrasi denganAWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan diAWS IoT Greengrass. CloudTrail menangkap semua panggilan API untuk AWS IoT Greengrass sebagai peristiwa. Panggilan yang direkam mencakup panggilan dari AWS IoT Greengrass konsol dan panggilan kode ke operasi API AWS IoT Greengrass ini. Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail acara secara berkelanjutan ke bucket Amazon S3, termasuk acara untuk. AWS IoT Greengrass Jika Anda tidak mengonfigurasi jejak, Anda masih dapat melihat peristiwa terbaru di CloudTrail konsol dalam Riwayat acara. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuatAWS IoT Greengrass, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk mempelajari selengkapnya CloudTrail, lihat Panduan AWS CloudTrail Pengguna.

AWS IoT Greengrassinformasi di CloudTrail

CloudTrail diaktifkan pada Akun AWS saat Anda membuat akun. Ketika aktivitas terjadi diAWS IoT Greengrass, aktivitas tersebut dicatat dalam suatu CloudTrail peristiwa bersama dengan peristiwa AWS layanan lainnya dalam riwayat Acara. Anda dapat melihat, mencari, dan mengunduh peristiwa terbaru di Akun AWS Anda. Untuk informasi selengkapnya, lihat Melihat peristiwa dengan riwayat CloudTrail acara.

Untuk catatan berkelanjutan tentang peristiwa di Akun AWS, termasuk peristiwa untuk AWS IoT Greengrass, buat jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Secara default, ketika Anda membuat jejak di dalam konsol tersebut, jejak diterapkan ke semua Wilayah AWSs. Jejak mencatat peristiwa dari semua Wilayah di partisi AWS dan mengirimkan

file log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi AWS layanan lain untuk menganalisis lebih lanjut dan menindaklanjuti data peristiwa yang dikumpulkan dalam CloudTrail log. Untuk informasi selengkapnya, lihat berikut:

- Gambaran umum untuk membuat jejak
- CloudTrail layanan dan integrasi yang didukung
- Mengonfigurasi notifikasi Amazon SNS untuk CloudTrail
- Menerima file CloudTrail log dari beberapa wilayah dan Menerima file CloudTrail log dari beberapa akun

Semua AWS IoT Greengrass tindakan dicatat oleh CloudTrail dan didokumentasikan dalam referensi AWS IoT Greengrass API. Misalnya, panggilan ke`AssociateServiceRoleToAccount`, `GetGroupVersionGetConnectivityInfo`, dan `CreateFunctionDefinition` tindakan menghasilkan entri dalam file CloudTrail log.

Setiap peristiwa atau entri log berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan berikut ini:

- Bahwa permintaan tersebut dibuat dengan kredensial pengguna root atau pengguna (IAM) AWS Identity and Access Management.
- Baik permintaan tersebut dibuat dengan kredensial keamanan sementara untuk peran atau pengguna gabungan.
- Apakah permintaan dibuat oleh layanan AWS lain.

Untuk informasi selengkapnya, lihat Elemen userIdentity CloudTrail .

Memahami entri file log AWS IoT Greengrass

Trail adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai file log ke bucket Amazon S3 yang Anda tentukan. CloudTrail file log berisi satu atau lebih entri log. Peristiwa mewakili permintaan tunggal dari sumber mana pun dan mencakup informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari panggilan API publik, sehingga file tersebut tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan `AssociateServiceRoleToAccount` tindakan.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/Mary_Major",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "Mary_Major"
    },
    "eventTime": "2018-10-17T17:04:02Z",
    "eventSource": "greengrass.amazonaws.com",
    "eventName": "AssociateServiceRoleToAccount",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "203.0.113.12",
    "userAgent": "apimanager.amazonaws.com",
    "errorCode": "BadRequestException",
    "requestParameters": null,
    "responseElements": {
        "Message": "That role ARN is invalid."
    },
    "requestID": "a5990ec6-d22e-11e8-8ae5-c7d2eEXAMPLE",
    "eventID": "b9070ce2-0238-451a-a9db-2dbf1EXAMPLE",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```


Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan `GetGroupVersion` tindakan.


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/Mary_Major",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "Mary_Major",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-10-17T18:14:57Z"
```

```
            }
        },
        "invokedBy": "apimanager.amazonaws.com"
    },
    "eventTime": "2018-10-17T18:15:11Z",
    "eventSource": "greengrass.amazonaws.com",
    "eventName": "GetGroupVersion",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "203.0.113.12",
    "userAgent": "apimanager.amazonaws.com",
    "requestParameters": {
        "GroupVersionId": "6c477753-dbf2-4cb8-acc3-5ba4eEXAMPLE",
        "GroupId": "90fcf6df-413c-4515-93a8-00056EXAMPLE"
    },
    "responseElements": null,
    "requestID": "95dcffce-d238-11e8-9240-a3993EXAMPLE",
    "eventID": "8a608034-82ed-431b-b5e0-87fbdEXAMPLE",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan `GetConnectivityInfo` tindakan.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/Mary_Major",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "Mary_Major"
    },
    "eventTime": "2018-10-17T17:02:12Z",
    "eventSource": "greengrass.amazonaws.com",
    "eventName": "GetConnectivityInfo",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "203.0.113.12",
    "userAgent": "apimanager.amazonaws.com",
    "requestParameters": {
        "ThingName": "us-east-1_CIS_1539795000000_"
```

```
    },
    "responseElements": null,
    "requestID": "63e3ebe3-d22e-11e8-9ddd-5baf3EXAMPLE",
    "eventID": "db2260d1-a8cc-4a65-b92a-13f65EXAMPLE",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan `CreateFunctionDefinition` tindakan.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/Mary_Major",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "Mary_Major"
    },
    "eventTime": "2018-10-17T18:01:11Z",
    "eventSource": "greengrass.amazonaws.com",
    "eventName": "CreateFunctionDefinition",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "203.0.113.12",
    "userAgent": "apimanager.amazonaws.com",
    "requestParameters": {
        "InitialVersion": "***"
    },
    "responseElements": {
        "CreationTimestamp": "2018-10-17T18:01:11.449Z",
        "LatestVersion": "dae06a61-c32c-41e9-b983-ee5cfEXAMPLE",
        "LatestVersionArn": "arn:aws:greengrass:us-east-1:123456789012:/greengrass/
definition/functions/7a94847d-d4d2-406c-9796-a3529EXAMPLE/versions/dae06a61-c32c-41e9-
b983-ee5cfEXAMPLE",
        "LastUpdatedTimestamp": "2018-10-17T18:01:11.449Z",
        "Id": "7a94847d-d4d2-406c-9796-a3529EXAMPLE",
        "Arn": "arn:aws:greengrass:us-east-1:123456789012:/greengrass/definition/
functions/7a94847d-d4d2-406c-9796-a3529EXAMPLE"
    },
    "requestID": "a17d4b96-d236-11e8-a74e-3db27EXAMPLE",
```

```
    "eventID": "bdbf6677-a47a-4c78-b227-c5f64EXAMPLE",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

Lihat juga

- Apa itu AWS CloudTrail? dalam AWS CloudTrail Panduan Penggguna
- Membuat EventBridge aturan yang memicu panggilan AWS API menggunakan CloudTrail dalam EventBridge Panduan Pengguna Amazon
- AWS IoT GreengrassReferensi API

Mengumpulkan data telemetri kondisi sistem dari perangkat AWS IoT Greengrass core

Data telemetri kondisi sistem adalah data diagnostik yang dapat membantu Anda memantau kinerja operasi kritis pada perangkat core Greengrass Anda. Agen telemetri pada core Greengrass mengumpulkan data telemetri lokal dan menerbitkannya ke Amazon EventBridge tanpa memerlukan interaksi pelanggan. Perangkat inti mempublikasikan data telemetri untuk EventBridge atas dasar upaya terbaik. Sebagai contoh, perangkat inti mungkin gagal untuk mengirimkan data telemetri saat offline.

> **Note**
>
> Amazon EventBridge adalah layanan bus peristiwa yang dapat Anda gunakan untuk menghubungkan aplikasi Anda dengan data dari berbagai sumber, seperti Perangkat core Greengrass and pemberitahuan deployment. Untuk informasi lebih lanjut, lihat Apa yang dimaksud dengan Amazon EventBridge? di Panduan EventBridge Pengguna Amazon.

Anda dapat membuat proyek dan aplikasi untuk mengambil, menganalisis, mengubah, dan melaporkan data telemetri dari perangkat edge Anda. Pakar domain, seperti proses engineer, dapat menggunakan aplikasi ini untuk mendapatkan wawasan tentang kondisi armada.

Untuk memastikan bahwa komponen edge Greengrass berfungsi dengan baik, AWS IoT Greengrass menggunakan data untuk tujuan pengembangan dan peningkatan kualitas. Fitur ini juga membantu

menginformasikan kemampuan edge yang baru dan ditingkatkan. AWS IoT Greengrass hanya menyimpan data telemetri hingga tujuh hari.

Fitur ini tersedia di perangkat lunak AWS IoT Greengrass core v1.11.0 dan diaktifkan secara default untuk semua core Greengrass, termasuk core yang ada. Anda secara otomatis mulai menerima data segera setelah Anda membarui ke perangkat lunak AWS IoT Greengrass core v1.11.0 atau yang lebih baru.

Untuk informasi tentang cara mengakses atau mengelola data telemetri yang dipublikasikan, lihat the section called “Berlangganan untuk menerima data telemetri”.

Agen telemetri mengumpulkan dan menerbitkan metrik sistem berikut.

Metrik Telemetri

Nama	Penjelasan	Sumber
`SystemMemUsage`	Jumlah memori yang saat ini digunakan oleh semua aplikasi pada perangkat inti Greengrass, termasuk sistem operasi.	Sistem
`CpuUsage`	Jumlah CPU yang saat ini digunakan oleh semua aplikasi pada perangkat inti Greengrass, termasuk sistem operasi.	Sistem
`TotalNumberOfFDs`	Bilangan deskriptor file yang disimpan oleh sistem operasi perangkat inti Greengrass. Satu file deskriptor secara unik mengidentifikasi satu file yang terbuka.	Sistem
`LambdaOutOfMemory`	Jumlah yang berjalan dan menghasilkan fungsi Lambda akan kehabisan memori.	Sistem

Nama	Penjelasan	Sumber
`DroppedMessageCount`	Jumlah pesan dibatalkan yang ditujukan untuk AWS IoT Core.	`GGCloudSpooler` komponen sistem
`LambdaTimeout`	Jumlah timeout untuk menjalankan fungsi Lambda yang ditetapkan pengguna.	Fungsi Lambda yang ditentukan pengguna, AWS Cloud, dan sistem
`LambdaUngracefullyKilled`	Jumlah yang berjalan yang ditetapkan pengguna fungsi Lambda gagal untuk selesai.	Fungsi Lambda yang ditentukan pengguna, AWS Cloud, dan sistem
`LambdaError`	Jumlah yang berjalan yang menghasilkan catatan kesalahan penulisan fungsi Lambda yang ditetapkan pengguna.	Fungsi Lambda yang ditentukan pengguna, AWS Cloud, dan sistem
`BytesAppended`	Jumlah byte data ditambahkan ke pengelola pengaliran.	`GGStreamManager` komponen sistem
`BytesUploadedToIoTAnalytics`	Jumlah byte data yang diekspor pengelola pengaliran ke saluran di AWS IoT Analytics.	`GGStreamManager` komponen sistem
`BytesUploadedToKinesis`	Jumlah byte data yang diekspor pengelola pengaliran ke pengaliran di Amazon Kinesis Data Streams.	`GGStreamManager` komponen sistem
`BytesUploadedToIoTSiteWise`	Jumlah byte data yang diekspor pengelola pengaliran ke properti aset di AWS IoT SiteWise.	`GGStreamManager` komponen sistem

Nama	Penjelasan	Sumber
`BytesUploadedToS3ExportTaskExecutor`	Jumlah byte data yang diekspor pengelola pengaliran ke objek di Amazon S3.	`GGStreamManager` komponen sistem
`BytesUploadedToHTTP`	Jumlah byte data yang diekspor pengelola pengaliran ke HTTP.	`GGStreamManager` komponen sistem

Mengonfigurasi pengaturan telemetri

Telemetri Greengrass menggunakan pengaturan berikut:

- Agen telemetri mengumpulkan data telemetri setiap jam.
- Agen telemetri menerbitkan pesan telemetri setiap 24 jam.

> **Note**
>
> Pengaturan tidak dapat diubah.

Anda dapat mengaktifkan atau menonaktifkan fitur telemetri untuk perangkat core Greengrass. AWS IoT Greengrass menggunakan bayangan untuk mengelola konfigurasi telemetri. Perubahan yang Anda lakukan berlaku segera ketika core memiliki koneksi ke AWS IoT Core.

Agen telemetri menerbitkan data menggunakan protokol MQTT dengan tingkat kualitas layanan (QoS) 0. Ini berarti bahwa hal tersebut tidak mengonfirmasi pengiriman atau mencoba lagi upaya penerbitan. Pesan telemetri berbagi koneksi MQTT dengan pesan lain untuk langganan yang ditujukan pada AWS IoT Core.

Selain biaya tautan data Anda, transfer data dari core ke AWS IoT Core tidak dipungut biaya. Hal ini karena agen menerbitkan ke topik AWS yang disimpan. Namun, tergantung pada kasus penggunaan Anda, Anda dapat dikenakan biaya saat menerima atau memproses data.

Persyaratan

Persyaratan berikut berlaku, ketika Anda mengonfigurasi pengaturan telemetri:

- Anda harus menggunakan perangkat lunak AWS IoT Greengrass core v1.11.0 atau yang lebih baru.

  > **Note**
  >
  > Jika Anda menjalankan versi sebelumnya dan Anda tidak ingin menggunakan telemetri, Anda tidak perlu melakukan apapun.

- Anda harus memberikan IAM izin untuk memperbarui core (hal) bayangan dan memanggil konfigurasi API sebelum Anda memperbarui pengaturan telemetri.

  Contoh kebijakan IAM berikut memungkinkan Anda mengelola bayangan dan waktu aktif konfigurasi core tertentu:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowManageShadow",
            "Effect": "Allow",
            "Action": [
                "iot:GetThingShadow",
                "iot:UpdateThingShadow",
                "iot:DeleteThingShadow",
                "iot:DescribeThing"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:thing/core-name-*"
            ]
        },
        {
            "Sid": "AllowManageRuntimeConfig",
            "Effect": "Allow",
            "Action": [
                "greengrass:GetCoreRuntimeConfiguration",
                "greengrass:UpdateCoreRuntimeConfiguration"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:thing/core-name"
            ]
        }
    ]
```

```
}
```

Anda dapat memberikan akses terperinci atau bersyarat ke sumber daya, misalnya, dengan menggunakan wildcard * skema penamaan. Untuk informasi lebih lanjut, lihat Penambahan dan Penghapusan kebijakan IAM dalam Panduan Pengguna IAM.

Konfigurasikan pengaturan telemetri (konsol)

Berikut ini menunjukkan cara memperbarui pengaturan telemetri core Greengrass di konsol AWS IoT Greengrass tersebut.

1. Di panel navigasiAWS IoT konsol, di bawah Kelola, luaskan perangkat Greengrass, lalu pilih Grup (V1).
2. Untuk grup Greengrass, pilih grup target.
3. Pada halaman konfigurasi grup, di bagian Ikhtisar, pilih inti Greengrass Anda.
4. Pada halaman konfigurasi core, pilih tab Telemetri.
5. Di bagian Telemetri kesehatan sistem, pilih Konfigurasi.
6. Di Konfigurasi telemetri, pilih Telemetri untuk mengaktifkan atau menonaktifkan status Telemetri.

> **Important**
>
> Secara default, fitur telemetri diaktifkan untuk perangkat lunak AWS IoT Greengrass core v1.11.0 atau yang lebih baru.

Perubahan berlaku pada saatwaktu aktif. Anda tidak perlu men-deploy grup.

Mengonfigurasi pengaturan telemetri (CLI)

Di API AWS IoT Greengrass tersebut, objek `TelemetryConfiguration` mewakili pengaturan telemetri core Greengrass. Objek ini adalah bagian dari objek `RuntimeConfiguration` yang terkait dengan core. Anda dapat menggunakan API AWS IoT Greengrass tersebut, AWS CLI, atau AWS SDK untuk mengelola telemetri Greengrass. Contoh dalam bagian ini menggunakanAWS CLI.

Untuk memeriksa pengaturan telemetri

Perintah berikut mendapat pengaturan telemetri dari core Greengrass.

- Ganti *core-thing-name*dengan nama core target.

  Untuk mendapatkan nama hal, Anda bisa menggunakan get-core-definition-versionperintah ini. Perintah mengembalikan ARN dari hal yang berisi nama hal.

```
aws greengrass get-thing-runtime-configuration --thing-name core-thing-name
```

Perintah mengembalikan objek `GetCoreRuntimeConfigurationResponse` dalam respons JSON. Misalnya:

```
{
    "RuntimeConfiguration": {
        "TelemetryConfiguration": {
            "ConfigurationSyncStatus": "OutOfSync",
            "Telemetry": "On"
        }
    }
}
```

Untuk mengonfigurasi pengaturan telemetri

Perintah berikut membarui pengaturan telemetri untuk core Greengrass.

- Ganti *core-thing-name*dengan nama core target.

  Untuk mendapatkan nama hal, Anda bisa menggunakan get-core-definition-versionperintah ini. Perintah mengembalikan ARN dari hal yang berisi nama hal.

JSON expanded

```
aws greengrass update-thing-runtime-configuration --thing-name core-thing-name --
telemetry-configuration  '{
    "RuntimeConfiguration": {
        "TelemetryConfiguration": {
            "ConfigurationSyncStatus": "InSync",
            "Telemetry": "Off"
        }
    }
}
```

JSON single-line

```
aws greengrass update-thing-runtime-configuration --thing-name core-thing-name --
telemetry-configuration "{\"TelemetryConfiguration\":{\"ConfigurationSyncStatus
\":\"InSync\",\"Telemetry\":\"Off\"}}"
```

Perubahan pengaturan telemetri telah diterapkan jika `ConfigurationSyncStatus` adalah `InSync`. Perubahan berlaku pada saatwaktu aktif. Anda tidak perlu men-deploy grup.

TelemetryConfiguration objek

Objek `TelemetryConfiguration` memiliki properti berikut:

`ConfigurationSyncStatus`

Periksa apakah pengaturan telemetri sudah sinkron. Anda mungkin tidak membuat perubahan pada properti ini.

Tipe: string

Nilai yang valid: `InSync` atau `OutOfSync`

`Telemetry`

Menghidupkan atau mematikan telemetri. Default-nya adalah `On`.

Tipe: string

Nilai yang valid: `On` atau `Off`

Berlangganan untuk menerima data telemetri

Anda dapat membuat aturan di Amazon EventBridge yang menentukan bagaimana memproses data telemetri agar diterbitkan dari perangkat inti Greengrass. Ketika EventBridge menerima data, ia memanggil tindakan target yang ditentukan dalam aturan Anda. Misalnya, Anda dapat membuat aturan acara yang mengirim notifikasi, menyimpan informasi acara, mengambil tindakan korektif, atau memanggil acra lain.

Peristiwa Telemetri

Acara untuk perubahan status deployment termasuk data telemetri menggunakan format berikut:

```
{
    "version": "0",
    "id": "f70f943b-9ae2-e7a5-fec4-4c22178a3e6a",
    "detail-type": "Greengrass Telemetry Data",
    "source": "aws.greengrass",
    "account": "123456789012",
    "time": "2020-07-28T20:45:53Z",
    "region": "us-west-1",
    "resources": [],
    "detail": {
        "ThingName": "CoolThing",
        "Schema": "2020-06-30",
        "ADP": [
            {
                "TS": 123231546,
                "NS": "StreamManager",
                "M": [
                    {
                        "N": "BytesAppended|BytesUploadedToKinesis",
                        "Sum": 11,
                        "U": "Bytes"
                    }
                ]
            },
            {
                "TS": 123231546,
                "NS": "StreamManager",
                "M": [
                    {
                        "N": "BytesAppended|BytesUploadedToS3ExportTaskExecutor",
                        "Sum": 11,
                        "U": "Bytes"
                    }
                ]
            },
            {
                "TS": 123231546,
                "NS": "StreamManager",
                "M": [
                    {
                        "N": "BytesAppended|BytesUploadedToHTTP",
                        "Sum": 11,
                        "U": "Bytes"
```

```
                }
            ]
        },
        {
            "TS": 123231546,
            "NS": "StreamManager",
            "M": [
                {
                    "N": "BytesAppended|BytesUploadedToIoTAnalytics",
                    "Sum": 11,
                    "U": "Bytes"
                }
            ]
        },
        {
            "TS": 123231546,
            "NS": "StreamManager",
            "M": [
                {
                    "N": "BytesAppended|BytesUploadedToIoTSiteWise",
                    "Sum": 11,
                    "U": "Bytes"
                }
            ]
        },
        {
            "TS": 123231546,
            "NS": "arn:aws:lambda:us-west-1:123456789012:function:my-function",
            "M": [
                {
                    "N": "LambdaTimeout",
                    "Sum": 15,
                    "U": "Count"
                }
            ]
        },
        {
            "TS": 123231546,
            "NS": "CloudSpooler",
            "M": [
                {
                    "N": "DroppedMessageCount",
                    "Sum": 15,
                    "U": "Count"
```

```
                }
            ]
        },
        {
            "TS": 1593727692,
            "NS": "SystemMetrics",
            "M": [
                {
                    "N": "SystemMemUsage",
                    "Sum": 11.23,
                    "U": "Megabytes"
                },
                {
                    "N": "CpuUsage",
                    "Sum": 35.63,
                    "U": "Percent"
                },
                {
                    "N": "TotalNumberOfFDs",
                    "Sum": 416,
                    "U": "Count"
                }
            ]
        },
        {
            "TS": 1593727692,
            "NS": "arn:aws:lambda:us-west-1:123456789012:function:my-function",
            "M": [
                {
                    "N": "LambdaOutOfMemory",
                    "Sum": 12,
                    "U": "Count"
                },
                {
                    "N": "LambdaUngracefullyKilled",
                    "Sum": 100,
                    "U": "Count"
                },
                {
                    "N": "LambdaError",
                    "Sum": 7,
                    "U": "Count"
                }
            ]
```

```
            }
        ]
    }
}
```

Array `ADP` berisi daftar titik data agregat yang memiliki sifat sebagai berikut:

`TS`

Diperlukan. Timestamp saat data dikumpulkan.

`NS`

Diperlukan. Namespace dari sistem.

`M`

Diperlukan. Daftar metrik. Metrik ini berisi properti berikut:

`N`

Nama metrik.

`Sum`

Nilai metrik agregat. Agen telemetri menambahkan nilai baru ke total sebelumnya, sehingga jumlahnya adalah nilai yang terus meningkat. Anda dapat menggunakan timestamp untuk menemukan nilai agregasi tertentu. Misalnya, untuk menemukan nilai agregat terbaru, kurangi nilai berstempel waktu sebelumnya dari nilai berstempel waktu terbaru.

`U`

Unit nilai metrik.

`ThingName`

Diperlukan. Nama perangkat hal yang Anda targetkan.

Prasyarat untuk membuat EventBridge aturan

Sebelum Anda membuat EventBridge aturan untukAWS IoT Greengrass, Anda harus melakukan hal berikut:

- Biasakan diri Anda dengan peristiwa, aturan, dan target di EventBridge.

- Buat dan konfigurasi target yang dipanggil oleh EventBridge aturan Anda. Aturan dapat memanggil berbagai jenis target, seperti Amazon Kinesis Streams, fungsi AWS Lambda, topik Amazon SNS, dan antrean Amazon SQS.

  EventBridge Aturan Anda, dan target terkait harus berada diWilayah AWS tempat Anda menciptakan sumber daya Greengrass Anda. Untuk informasi selengkapnya, lihat Endpoint dan kuota layanan di bagian Referensi Umum AWS.

Untuk informasi lebih lanjut, lihat Apa yang dimaksud dengan Amazon EventBridge? dan Memulai dengan Amazon EventBridge di Panduan EventBridge Pengguna Amazon.

Buat aturan peristiwa untuk mendapatkan data telemetri (konsol)

Gunakan langkah-langkah berikut untuk menggunakan dalamAWS Management Console membuat EventBridge aturan yang menerima data telemetri agar diterbitkan oleh core Greengrass. Hal ini memungkinkan server web, alamat email, dan pelanggan topik lainnya untuk menanggapi peristiwa tersebut. Untuk informasi selengkapnya, lihat Membuat EventBridge aturan yang memicu peristiwa dariAWS sumber daya di Panduan EventBridge Pengguna Amazon.

1. Buka EventBridgekonsol Amazon dan pilih Buat aturan.
2. Di bawah Nama dan deskripsi, masukkan nama dan deskripsi untuk alarm Anda.
3. Pilih bus acara - dan aktifkan aturan pada bus acara yang dipilih..
4. Pilih jenis Aturan dan pilih Aturan dengan pola peristiwa.
5. Pilih Selanjutnya.
6. Untuk Sumber acara, pilih AWSacara atau acara EventBridge mitra.
7. Untuk contoh acara, pilih AWSacara, dan pilih Greengrass Telemetri Data.
8. Dalam pola acara, membuat pilihan berikut:

   a. Untuk sumber acara, pilih AWSlayanan.
   b. Untuk AWSlayanan, pilih Greengrass.
   c. Untuk Jenis acara, pilih Data Telemetri Greengrass.
9. Pilih Selanjutnya.
10. Untuk Target 1, pilih AWSlayanan.
11. Untuk Pilih target, pilih antrian SQS.
12. Untuk Antrian, pilih fungsi Anda.

Buat aturan peristiwa untuk mendapatkan data telemetri (CLI)

Gunakan langkah-langkah berikut untuk menggunakan dalamAWS CLI membuat EventBridge aturan yang menerima data telemetri agar diterbitkan oleh core Greengrass. Hal ini memungkinkan server web, alamat email, dan pelanggan topik lainnya untuk menanggapi peristiwa tersebut.

1. Buat aturan .

   - Ganti *nama-hal* dengan nama core.

     Untuk mendapatkan nama hal, Anda bisa menggunakan get-core-definition-versionperintah ini. Perintah mengembalikan ARN dari hal yang berisi nama hal.

   ```
   aws events put-rule \
     --name TestRule \
     --event-pattern "{\"source\": [\"aws.greengrass\"], \"detail\": {\"ThingName\":
    [\"thing-name\"]}}"
   ```

   Properti apa pun yang Anda hilangkan dari pola akan diabaikan.

2. Tambahkan topik sebagai target aturan. Contoh berikut menggunakan Amazon SQS tetapi Anda dapat mengonfigurasi jenis target lainnya.

   - Ganti *queue-arn* dengan ARN dari antrean Amazon SQS Anda.

   ```
   aws events put-targets \
     --rule TestRule \
     --targets "Id"="1","Arn"="queue-arn"
   ```

   > **Note**
   >
   > Untuk mengizinkan Amazon EventBridge untuk memanggil antrean target Anda, Anda harus menambahkan kebijakan berbasis sumber daya pada topik Anda. Untuk informasi selengkapnya, lihat izin Amazon SQS di Panduan EventBridge Pengguna Amazon.

Untuk informasi selengkapnya, lihat Pola peristiwa dan peristiwa EventBridge di dalam Panduan EventBridge Pengguna Amazon.

Penyelesaian Masalah AWS IoT Greengrass telemetri

Gunakan informasi berikut untuk menyelesaikan masalah dengan mengonfigurasi AWS IoT Greengrass telemetri.

Error: Respons berisi "ConfigurationStatus“:OutOfSync" "setelah Anda menjalankan get-thing-runtime-configuration perintah

Solusi:

- Layanan Bayangan Perangkat AWS IoT membutuhkan waktu untuk memproses pembaruan konfigurasi waktu aktif dan untuk memberikan pembaruan ke perangkat core Greengrass. Anda dapat menunggu dan memeriksa apakah pengaturan telemetri nanti akan disinkronkan.
- Pastikan bahwa perangkat core Anda online.
- Mengaktifkan Amazon CloudWatch Logs diAWS IoT Core untuk memantau bayangannya.
- Gunakan AWS IoT metrik untuk memantau hal Anda.

Memanggil API pemeriksaan kondisi lokal

AWS IoT Greengrass berisi API HTTP lokal yang menyediakan snapshot dari keadaan saat ini dari proses pekerja lokal yang dimulai oleh AWS IoT Greengrass. Snapshot ini mencakup fungsi Lambda yang ditetapkan pengguna dan fungsi Lambda sistem. Fungsi Lambda sistem adalah bagian dari perangkat lunak AWS IoT Greengrass core. Mereka berjalan sebagai pekerja lokal yang memproses pada perangkat core dan mengelola operasi seperti perutean pesan, sinkronisasi bayangan lokal, dan deteksi alamat IP otomatis.

API pemeriksaan kondisi support permintaan berikut:

- Mengirim permintaan `GET` untuk mendapatkan informasi kondisi dari semua pekerja.
- Mengirim permintaan `POST` untuk mendapatkan informasi kondisi dari pekerja tertentu.

Permintaan dikirim secara lokal memalui perangkat dan tidak memerlukan koneksi internet.

Dapatkan informasi kondisi untuk semua pekerja

Mengirim `GET` permintaan untuk mendapatkan informasi kondisi tentang semua pekerja yang sedang bekerja.

- Ganti *Port* dengan nomor port IPC.

```
GET http://localhost:port/2016-11-01/health/workers
```

`port`

Nomor port IPC.

Nilai dapat bervariasi antara 1024 dan 65535. Nilai default-nya adalah 8000.

Untuk mengubah nomor port ini, Anda dapat memperbarui properti `ggDaemonPort` pada file `config.json` tersebut. Untuk informasi selengkapnya, lihat AWS IoT Greengrass file konfigurasi core.

Contoh Permintaan

Contoh permintaan `curl` berikut untuk mendapatkan informasi kondisi dari semua pekerja.

```
curl http://localhost:8000/2016-11-01/health/workers
```

Tanggapan JSON

Permintaan ini mengembalikan array dari objek informasi kondisi pekerja ini.

Contoh tanggapan

Contoh respons berikut mencantumkan informasi kondisi objek dari semua proses pekerja yang dimulai oleh AWS IoT Greengrass.

```
[
    {
      "FuncArn": "arn:aws:lambda:::function:GGShadowService:1",
      "WorkerId" : "65515053-2f70-43dc-7cc0-1712bEXAMPLE",
      "ProcessId": "1234",
      "WorkerState": "Waiting"
    },
    {
      "FuncArn": "arn:aws:lambda:::function:GGSecretManager:1",
      "WorkerId": "a9916cc2-1b4d-4f0e-4b12-b1872EXAMPLE",
      "ProcessId": "9798",
```

```
      "WorkerState": "Waiting"
    },
    {
      "FuncArn": "arn:aws:lambda:us-west-2:123456789012:function:my-lambda-function:3",
      "WorkerId": "2e6f785e-66a5-42c9-67df-42073EXAMPLE",
      "ProcessId": "11837",
      "WorkerState": "Waiting"
    },
    ...
]
```

Dapatkan informasi kondisi tentang pekerja tertentu

Mengirim permintaan POST untuk mendapatkan informasi kondisi dari pekerja tertentu. Ganti *Port* dengan nomor port IPC. Default-nya adalah 8000.

```
POST http://localhost:port/2016-11-01/health/workers
```

Contoh Permintaan

Contoh permintaan berikut `curl` untuk mendapatkan informasi kondisi dari pekerja tertentu.

```
curl --data "@body.json" http://localhost:8000/2016-11-01/health/workers
```

Inilah contoh `body.json` isi permintaan:

```
{
    "FuncArns": [
        "arn:aws:lambda:::function:GGShadowService:1",
        "arn:aws:lambda:us-west-2:123456789012:function:my-lambda-function:3"
    ]
}
```

Isi permintaan berisi `FuncArns` array.

`FuncArns`

Daftar Amazon Resource Name (ARN) untuk fungsi Lambda yang mewakili pekerja target.

- Untuk fungsi Lambda yang ditetapkan pengguna, tentukan ARN versi yang digunakan saat ini. Jika Anda menambahkan fungsi Lambda ke grup menggunakan alias ARN, Anda dapat

menggunakan permintaan GET untuk mendapatkan semua data pekerja, dan kemudian memilih ARN yang Anda inginkan untuk kueri.

- Untuk fungsi Lambda sistem, tentukan ARN dari fungsi Lambda yang sesuai. Untuk informasi selengkapnya, lihat the section called “Fungsi Lambda sistem”.

Jenis: array string

Panjang minimum: 1

Panjang maksimum: Jumlah total pekerja yang dimulai olehAWS IoT Greengrasspada perangkat inti.

Tanggapan JSON

Permintaan ini mengembalikan `Workers` array dan `InvalidArns` array.

`Workers`

Daftar objek informasi kondisi untuk pekerja tertentu.

Jenis: array dari objek informasi kondisi

`InvalidArns`

Daftar fungsi ARN yang tidak valid, termasuk fungsi ARN yang tidak memiliki pekerja terkait.

Jenis: array string

Contoh tanggapan

Berikut daftar contoh respons objek informasi kondisi untuk pekerja tertentu.

```
{
    "Workers": [
        {
            "FuncArn": "arn:aws:lambda:::function:GGShadowService:1",
            "WorkerId" : "65515053-2f70-43dc-7cc0-1712bEXAMPLE",
            "ProcessId": "1234",
            "WorkerState": "Waiting"
        },
        {
```

```
            "FuncArn": "arn:aws:lambda:us-west-2:123456789012:function:my-lambda-
function:3",
            "WorkerId": "2e6f785e-66a5-42c9-67df-42073ESAMPLE",
            "ProcessId": "11837",
            "WorkerState": "Waiting"
        }
    ],
    "InvalidArns" : [
        "some-malformed-arn",
        "arn:aws:lambda:us-west-2:123456789012:function:some-unknown-function:1"
    ]
}
```

Permintaan ini mengembalikan error berikut:

400 Permintaan tidak valid

Isi permintaan yang salah format. Untuk mengatasi masalah ini, gunakan format berikut dan kirim permintaan:

```
{"FuncArns":["function-1-arn","function-2-arn"]}
```

400 Permintaan melebihi jumlah pekerja maksimum

Jumlah ARN yang ditentukan dalam array `FuncArns` melebihi jumlah pekerja.

Informasi kondisi pekerja

Objek informasi kondisi berisi properti berikut:

`FuncArn`

ARN dari fungsi Lambda sistem yang mewakili pekerja.

Tipe: `string`

`WorkerId`

ID pekerja. Properti ini dapat berguna untuk debugging. File `runtime.log` dan catatan fungsi Lambda berisi ID pekerja, sehingga properti ini dapat sangat berguna untuk debug fungsi Lambda sesuai permintaan yang menjalankan beberapa instans.

Tipe: `string`

`ProcessId`

Proses ID (PID) dari proses pekerja.

Tipe: `int`

`WorkerState`

Keadaan pekerja.

Tipe: `string`

Berikut ini adalah kemungkinan status pekerja:

`Working`

Memproses pesan.

`Waiting`

Menunggu pesan. Berlaku untuk fungsi Lambda berumur panjang yang berjalan sebagai daemon atau proses mandiri.

`Starting`

Berputar, memulai.

`FailedInitialization`

Gagal menginisialisasi.

`Terminated`

Dihentikan oleh daemon Greengrass

`NotStarted`

Gagal memulai, membuat upaya memulai lagi.

`Initialized`

Berhasil diinisialisasi.

Fungsi Lambda sistem

Anda dapat meminta informasi kondisi untuk fungsi Lambda sistem berikut:

`GGCloudSpooler`

Mengelola antrean untuk pesan MQTT yang memiliki AWS IoT Core sebagai sumber atau target.

ARN:`arn:aws:lambda:::function:GGCloudSpooler:1`

`GGConnManager`

Rute pesan MQTT antara core Greengrass dan perangkat klien.

ARN:`arn:aws:lambda:::function:GGConnManager`

`GGDeviceCertificateManager`

MendengarkanAWS IoTbayangan untuk perubahan core titik akhir IP dan menghasilkan sertifikat server yang digunakan oleh GGConnManager untuk saling autentikasi.

ARN:`arn:aws:lambda:::function:GGDeviceCertificateManager`

`GGIPDetector`

Mengelola deteksi alamat IP otomatis yang memungkinkan perangkat dalam grup Greengrass untuk menemukan perangkat core Greengrass. Layanan ini tidak berlaku bila Anda memberikan alamat IP secara manual.

ARN:`arn:aws:lambda:::function:GGIPDetector:1`

`GGSecretManager`

Mengelola penyimpanan aman dari rahasia lokal dan akses oleh Lambda dan konektor yang ditentukan pengguna.

ARN:`arn:aws:lambda:::function:GGSecretManager:1`

`GGShadowService`

Mengelola bayangan lokal untuk perangkat klien.

ARN:`arn:aws:lambda:::function:GGShadowService`

`GGShadowSyncManager`

Menyinkronkan bayangan lokal denganAWS Clouduntuk perangkat inti dan perangkat klien, jika perangkat`syncShadow`properti diatur ke`true`.

ARN:`arn:aws:lambda:::function:GGShadowSyncManager`

`GGStreamManager`

Memproses aliran data secara lokal dan melakukan ekspor otomatis ke AWS Cloud.

ARN:`arn:aws:lambda:::function:GGStreamManager:1`

`GGTES`

Layanan pertukaran token lokal yang mengambil kredensial IAM didefinisikan dalam peran grup Greengrass yang menggunakan kode lokal untuk mengakses layanan AWS ini.

ARN:`arn:aws:lambda:::function:GGTES`

Menandai Sumber Daya AWS IoT Greengrass Anda

Tanda dapat membantu Anda mengatur dan mengelola grup AWS IoT Greengrass Anda. Anda dapat menggunakan tanda untuk menugaskan metadata ke grup, deployment massal, dan core, perangkat, dan sumber daya lainnya yang ditambahkan ke grup. Tanda juga dapat digunakan dalam kebijakan IAM untuk menentukan akses bersyarat ke sumber daya Greengrass Anda.

> **Note**
>
> Sekarang, Tanda sumber daya Greengrass tidak didukung untuk grup penagihan AWS IoT atau laporan alokasi biaya.

Dasar-dasar tanda

Tanda mengizinkan Anda untuk mengategorikan sumber daya AWS IoT Greengrass Anda, sebagai contoh, berdasarkan tujuan, pemilik, atau lingkungan. Saat Anda memiliki banyak sumber daya dengan jenis yang sama, Anda dapat dengan segera mengidentifikasi sumber daya berdasarkan tanda yang terlampir. Setiap tanda terdiri dari kunci dan nilai opsional, yang keduanya Anda tentukan. Kami menyarankan agar Anda merancang serangkaian kunci tanda yang konsisten untuk setiap jenis sumber daya. Penggunaan set kunci tanda yang konsisten akan memudahkan pengelolaan sumber daya Anda. Sebagai contoh, Anda dapat menentukan serangkaian tanda untuk grup Anda yang dapat membantu Anda melacak lokasi pabrik perangkat core Anda. Untuk mengetahui informasi lebih lanjut, lihat AWS Strategi Penandaan.

Dukungan penandaan di konsol AWS IoT tersebut

Anda dapat membuat, melihat, dan mengelola tanda pada Greengrass Sumber daya `Group` Anda di konsol AWS IoT tersebut. Sebelum Anda membuat tanda, perhatikan pembatasan penandaan. Untuk informasi selengkapnya, lihat Konvensi penamaan dan penggunaan tag di. Referensi Umum Amazon Web Services

Untuk menetapkan tag saat Anda membuat grup

1. Anda dapat menetapkan tanda ke grup saat Anda membuat grup. Pilih Tambahkan tag baru di bawah bagian Tag untuk menampilkan kolom input penandaan.

Untuk melihat dan mengelola tag dari halaman konfigurasi grup

Anda dapat melihat dan mengelola tag dari halaman konfigurasi grup dengan memilih Pengaturan tampilan. Di bagian Tag untuk grup, pilih Kelola tag untuk menambahkan, mengedit, atau menghapus tag grup.

Dukungan penandaan di API AWS IoT Greengrass ini

Anda dapat menggunakan API AWS IoT Greengrass untuk membuat, mendaftar, dan mengelola tanda pada AWS IoT Greengrass sumber daya yang mendukung penandaan. Sebelum Anda membuat tag, perhatikan pembatasan penandaan. Untuk informasi selengkapnya, lihat Konvensi penamaan dan penggunaan tag di. Referensi Umum Amazon Web Services

- Untuk menambahkan tanda selama pembuatan sumber daya, definisikan mereka dalam properti `tags` dari sumber daya.
- Untuk menambahkan tanda setelah sumber daya dibuat, atau untuk memperbarui nilai tanda, gunakan tindakan `TagResource` ini.
- Untuk menghapus tanda dari sumber daya, gunakan tindakan `UntagResource` ini.
- Untuk mengambil tanda yang terkait dengan sumber daya, gunakan tindakan `ListTagsForResource` atau dapatkan sumber daya dan periksa properti `tags` ini.

Tabel berikut mencantumkan sumber daya yang dapat Anda tandai di API AWS IoT Greengrass dan yang sesuai untuk `Create` dan tindakan `Get` ini.

Sumber daya	Buat	Dapatkan
`Group`	`CreateGroup`	`GetGroup`
`ConnectorDefinition`	`CreateConnectorDefinition`	`GetConnectorDefinition`
`CoreDefinition`	`CreateCoreDefinition`	`GetCoreDefinition`
`DeviceDefinition`	`CreateDeviceDefinition`	`GetDeviceDefinition`

Sumber daya	Buat	Dapatkan
FunctionDefinition	CreateFunctionDefinition	GetFunctionDefinition
LoggerDefinition	CreateLoggerDefinition	GetLoggerDefinition
ResourceDefinition	CreateResourceDefinition	GetResourceDefinition
SubscriptionDefinition	CreateSubscriptionDefinition	GetSubscriptionDefinition
BulkDeployment	StartBulkDeployment	GetBulkDeploymentStatus

Gunakan tindakan berikut untuk mencantumkan dan mengelola tanda pada sumber daya yang mendukung penandaan:

- TagResource. Menambahkan tanda ke sumber daya. Juga digunakan untuk mengubah nilai pasangan nilai kunci tanda ini.
- ListTagsForResource. Daftar tanda untuk sumber daya.
- UntagResource. Menghapus tanda dari sumber daya.

Anda dapat menambahkan atau menghapus tanda pada sumber daya kapan saja. Untuk mengubah nilai kunci tanda, menambahkan tanda ke sumber daya yang mendefinisikan kunci yang sama dan nilai baru. Nilai baru menimpa nilai lama. Anda dapat mengatur nilai tanda menjadi sebuah string kosong, tetapi Anda tidak dapat mengatur nilai tanda menjadi nol.

Jika Anda menghapus sebuah sumber daya, semua tanda yang berkaitan dengan sumber daya tersebut juga dihapus.

> **Note**
>
> Jangan bingung antara tanda sumber daya dengan atribut yang dapat Anda tetapkan pada hal AWS IoT ini. Meskipun core Greengrass adalah hal AWS IoT ini, tanda sumber daya yang dijelaskan dalam topik ini dilampirkan ke `CoreDefinition`, bukan ke core.

Menggunakan tanda dengan kebijakan IAM

Di kebijakan IAM, Anda dapat menggunakan tag sumber daya untuk mengontrol akses pengguna dan izin. Misalnya, kebijakan dapat mengizinkan pengguna untuk membuat hanya sumber daya yang memiliki tanda tertentu. Kebijakan juga dapat membatasi pengguna untuk membuat atau memodifikasi sumber daya yang memiliki tanda tertentu. Anda dapat memberi tanda pada sumber daya selama pembuatan (disebut Tag dibuat) sehingga nanti Anda tidak perlu menjalankan skrip penandaan khusus. Ketika lingkungan baru diluncurkan dengan tanda, izin IAM yang sesuai akan diterapkan secara otomatis.

Kunci konteks syarat berikut dan nilai-nilai dapat digunakan dalam elemen `Condition` (juga disebut `Condition` blok) dari kebijakan.

`greengrass:ResourceTag/`*`tag-key`*`: `*`tag-value`*

Izinkan atau tolak tindakan pengguna pada sumber daya dengan tanda tertentu.

`aws:RequestTag/`*`tag-key`*`: `*`tag-value`*

Memerlukan tanda tertentu untuk digunakan (atau tidak digunakan) saat membuat permintaan API agar membuat atau memodifikasi tanda pada sumber daya yang dapat ditandai.

`aws:TagKeys: [`*`tag-key`*`, ...]`

Memerlukan sekumpulan kunci tanda tertentu untuk digunakan (atau tidak digunakan) saat membuat permintaan API agar membuat atau memodifikasi sumber daya yang dapat ditandai.

Syarat kunci konteks dan nilai-nilai dapat digunakan hanya pada tindakan AWS IoT Greengrass yang bertindak pada sumber daya yang dapat ditandai. Tindakan ini mengambil sumber daya sebagai parameter yang diperlukan. Sebagai contoh, Anda dapat mengatur akses bersyarat pada `GetGroupVersion`. Anda tidak dapat mengatur akses bersyarat di `AssociateServiceRoleToAccount` karena tidak ada sumber daya (sebagai contoh, grup, definisi core, atau definisi perangkat) yang dapat ditandai untuk direferensikan dalam permintaan.

Untuk informasi selengkapnya, lihat Mengontrol akses menggunakan tanda dan Referensi kebijakan IAM JSON dalam Panduan Pengguna IAM. Referensi kebijakan JSON mencakup sintaks detail, deskripsi, dan contoh elemen, variabel, dan logika evaluasi kebijakan JSON di IAM.

Contoh kebijakan IAM

Kebijakan contoh berikut menerapkan izin berbasis tanda yang membatasi pengguna beta untuk tindakan pada sumber daya beta saja.

- Pernyataan pertama mengizinkan pengguna IAM untuk bertindak pada sumber daya yang memiliki tanda env=beta saja.
- Pernyataan kedua mencegah pengguna IAM menghapus tanda env=beta dari sumber daya. Ini melindungi pengguna dari menghapus akses mereka sendiri.

> **Note**
>
> Jika Anda menggunakan tanda untuk mengontrol akses ke sumber daya, Anda juga harus mengelola izin yang memungkinkan pengguna untuk menambahkan tanda atau menghapus tanda dari sumber daya yang sama tersebut. Jika tidak, dalam beberapa kasus, pengguna dapat mengakali pembatasan Anda dan mendapatkan akses atas sumber daya dengan memodifikasi tandanya.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "greengrass:*",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "greengrass:ResourceTag/env": "beta"
                }
            }
        },
        {
            "Effect": "Deny",
            "Action": "greengrass:UntagResource",
            "Resource": "*",
```

```
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/env": "beta"
                }
            }
        }
    ]
}
```

Untuk mengizinkan pengguna menandai pada buat, Anda harus memberinya izin yang sesuai. Kebijakan berikut mencakup syarat `"aws:RequestTag/env": "beta"` pada tindakan `greengrass:TagResource` dan `greengrass:CreateGroup` ini, yang memungkinkan pengguna untuk membuat grup hanya jika mereka menandai grup dengan env=beta. Hal ini secara efektif memaksa pengguna untuk menandai grup baru.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "greengrass:TagResource",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/env": "beta"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "greengrass:CreateGroup",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/env": "beta"
                }
            }
        }
    ]
}
```

Potongan berikut menunjukkan bagaimana Anda dapat menentukan beberapa nilai tanda untuk kunci tanda dengan melampirkannya dalam daftar:

```
"StringEquals" : {
    "greengrass:ResourceTag/env" : ["dev", "test"]
}
```

Lihat juga

- Menandai AWS sumber daya di Referensi Umum Amazon Web Services

AWS CloudFormation dukungan untuk AWS IoT Greengrass

AWS CloudFormation ini adalah layanan yang dapat membantu Anda membuat, mengatur, and menjawab sumber daya AWS Anda. Anda dapat menggunakan AWS CloudFormation template untuk menentukan AWS IoT Greengrass grup dan perangkat klien, langganan, dan komponen lain yang ingin Anda terapkan. Lihat contohnya di the section called “Contoh Templat”.

Sumber daya dan infrastruktur yang Anda hasilkan dari templat disebut tumpukan. Anda dapat menentukan semua sumber daya Anda dalam satu templat atau merujuk ke sumber daya dari tumpukan lainnya. Untuk informasi lebih lanjut tentang templat AWS CloudFormation dan fitur, lihat Apa AWS CloudFormation? di AWS CloudFormation Panduan Pengguna.

Membuat sumber daya

AWS CloudFormation templat adalah dokumen JSON atau YAML yang menggambarkan sifat dan hubungan AWS sumber daya. Berikut AWS IoT Greengrass sumber daya yang didukung:

- Grup
- Core
- Perangkat klien (perangkat)
- Fungsi Lambda
- Konektor
- Sumber daya (lokal, machine learning, dan rahasia)
- Berlangganan
- Logger (logging konfigurasi)

Masuk AWS CloudFormation templat, struktur dan sintaks dari sumber daya Greengrass didasarkan pada AWS IoT Greengrass API. Misalnya, contoh template mengaitkan tingkat atas `DeviceDefinition` dengan `DeviceDefinitionVersion` yang berisi perangkat klien individual. Untuk informasi selengkapnya, lihat the section called “Ringkasan tentang model objek grup”.

Pada AWS IoT Greengrass referensi jenis sumber daya di AWS CloudFormation Panduan Pengguna menjelaskan sumber daya Greengrass yang dapat Anda kelola dengan AWS CloudFormation. Saat Anda menggunakan AWS CloudFormation templat untuk membuat sumber daya Greengrass, kami sarankan Anda mengelola mereka hanya dari AWS CloudFormation. Sebagai contoh, Anda harus

memperbarui templat Anda jika Anda ingin menambahkan, mengubah, atau menghapus perangkat (bukan menggunakan AWS IoT Greengrass API atau AWS IoT Konsol). Hal ini memungkinkan Anda untuk menggunakan rollback dan lainnya AWS CloudFormation mengubah fitur manajemen. Untuk informasi lebih lanjut tentang penggunaan AWS CloudFormation untuk membuat dan mengelola sumber daya dan tumpukan Anda, lihat Bekerja dengan tumpukan di AWS CloudFormation Panduan Pengguna.

Untuk panduan yang menunjukkan cara membuat dan menerapkan AWS IoT Greengrass sumber daya dalam AWS CloudFormation templat, lihat Mengotomatisasi AWS IoT Greengrass pengaturan dengan AWS CloudFormation di Internet untuk Segala (IoT) pada AWS Blog Resmi.

Menerapkan sumber daya

Setelah Anda membuat AWS CloudFormation tumpukan yang berisi versi grup Anda, Anda dapat menggunakan AWS CLI atau AWS IoT konsol untuk menerapkan itu.

> **Note**
>
> Untuk men-deploy grup, Anda harus memiliki peran layanan Greengrass yang terkait dengan Akun AWS. Peran layanan membolehkan AWS IoT Greengrass untuk mengakses sumber daya Anda di AWS Lambda dan layanan AWS lainnya. Peran ini harus ada jika Anda sudah menerapkan grup Greengrass di saat ini Wilayah AWS. Untuk informasi selengkapnya, lihat the section called “Peran layanan Greengrass”.

Untuk menyebarkan grup () AWS CLI

- Jalankan create-deployment perintah.

  ```
  aws greengrass create-deployment --group-id GroupId --group-version-
  id GroupVersionId --deployment-type NewDeployment
  ```

  > **Note**
  >
  > Pernyataan `CommandToDeployGroup` di contoh templat menunjukkan cara mengeluarkan perintah dengan ID versi grup dan grup saat Anda membuat tumpukan.

Untuk menyebarkan grup (konsol)

1. Di panel navigasi AWS IoT konsol, di bawah Kelola, perluas perangkat Greengrass, lalu pilih Grup (V1).
2. Pilih grup Anda.
3. Pada halaman konfigurasi grup, pilih Deploy.

Contoh Templat

Contoh template berikut membuat grup Greengrass yang berisi inti, perangkat klien, fungsi, logger, langganan, dan dua sumber daya. Untuk melakukannya, templat mengikuti model objek AWS IoT Greengrass API. Misalnya, perangkat klien yang ingin Anda tambahkan ke grup terkandung dalam `DeviceDefinitionVersion` sumber daya, yang terkait dengan `DeviceDefinition` sumber daya. Untuk menambahkan perangkat ke grup, versi grup referensi ARN `DeviceDefinitionVersion`.

Templat meliputi parameter yang memungkinkan Anda menentukan sertifikat ARN untuk core dan perangkat dan versi ARN dari fungsi Lambda sumber (yang merupakan AWS Lambda Sumber daya). Itu menggunakan `Ref` dan `GetAtt` fungsi intrinsik untuk referensi ID, ARN, dan atribut lain yang diperlukan untuk membuat sumber daya Greengrass.

Template juga mendefinisikan dua AWS IoT perangkat (benda), yang mewakili perangkat inti dan klien yang ditambahkan ke grup Greengrass.

Setelah Anda membuat tumpukan dengan sumber daya Greengrass Anda, Anda dapat menggunakan AWS CLI atau Konsol AWS IoT tersebut ke men-deploy grup.

> **Note**
>
> Pernyataan `CommandToDeployGroup` dalam contoh menunjukkan bagaimana output lengkap create-deployment perintah CLI yang dapat Anda gunakan untuk menerapkan grup Anda.

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
```

```
   "Description": "AWS IoT Greengrass example template that creates a group version
with a core, device, function, logger, subscription, and resources.",
   "Parameters": {
       "CoreCertificateArn": {
           "Type": "String"
       },
       "DeviceCertificateArn": {
           "Type": "String"
       },
       "LambdaVersionArn": {
           "Type": "String"
       }
   },
   "Resources": {
       "TestCore1": {
           "Type": "AWS::IoT::Thing",
           "Properties": {
               "ThingName": "TestCore1"
           }
       },
       "TestCoreDefinition": {
           "Type": "AWS::Greengrass::CoreDefinition",
           "Properties": {
               "Name": "DemoTestCoreDefinition"
           }
       },
       "TestCoreDefinitionVersion": {
           "Type": "AWS::Greengrass::CoreDefinitionVersion",
           "Properties": {
               "CoreDefinitionId": {
                   "Ref": "TestCoreDefinition"
               },
               "Cores": [
                   {
                       "Id": "TestCore1",
                       "CertificateArn": {
                           "Ref": "CoreCertificateArn"
                       },
                       "SyncShadow": "false",
                       "ThingArn": {
                           "Fn::Join": [
                               ":",
                               [
                                   "arn:aws:iot",
```

```
                                {
                                    "Ref": "AWS::Region"
                                },
                                {
                                    "Ref": "AWS::AccountId"
                                },
                                "thing/TestCore1"
                            ]
                        ]
                    }
                }
            ]
        }
    },
    "TestClientDevice1": {
        "Type": "AWS::IoT::Thing",
        "Properties": {
            "ThingName": "TestClientDevice1"
        }
    },
    "TestDeviceDefinition": {
        "Type": "AWS::Greengrass::DeviceDefinition",
        "Properties": {
            "Name": "DemoTestDeviceDefinition"
        }
    },
    "TestDeviceDefinitionVersion": {
        "Type": "AWS::Greengrass::DeviceDefinitionVersion",
        "Properties": {
            "DeviceDefinitionId": {
                "Fn::GetAtt": [
                    "TestDeviceDefinition",
                    "Id"
                ]
            },
            "Devices": [
                {
                    "Id": "TestClientDevice1",
                    "CertificateArn": {
                        "Ref": "DeviceCertificateArn"
                    },
                    "SyncShadow": "true",
                    "ThingArn": {
                        "Fn::Join": [
```

```
                            ":",
                            [
                                "arn:aws:iot",
                                {
                                    "Ref": "AWS::Region"
                                },
                                {
                                    "Ref": "AWS::AccountId"
                                },
                                "thing/TestClientDevice1"
                            ]
                        ]
                    }
                }
            ]
        }
    },
    "TestFunctionDefinition": {
        "Type": "AWS::Greengrass::FunctionDefinition",
        "Properties": {
            "Name": "DemoTestFunctionDefinition"
        }
    },
    "TestFunctionDefinitionVersion": {
        "Type": "AWS::Greengrass::FunctionDefinitionVersion",
        "Properties": {
            "FunctionDefinitionId": {
                "Fn::GetAtt": [
                    "TestFunctionDefinition",
                    "Id"
                ]
            },
            "DefaultConfig": {
                "Execution": {
                    "IsolationMode": "GreengrassContainer"
                }
            },
            "Functions": [
                {
                    "Id": "TestLambda1",
                    "FunctionArn": {
                        "Ref": "LambdaVersionArn"
                    },
                    "FunctionConfiguration": {
```

```
                        "Pinned": "true",
                        "Executable": "run.exe",
                        "ExecArgs": "argument1",
                        "MemorySize": "512",
                        "Timeout": "2000",
                        "EncodingType": "binary",
                        "Environment": {
                            "Variables": {
                                "variable1": "value1"
                            },
                            "ResourceAccessPolicies": [
                                {
                                    "ResourceId": "ResourceId1",
                                    "Permission": "ro"
                                },
                                {
                                    "ResourceId": "ResourceId2",
                                    "Permission": "rw"
                                }
                            ],
                            "AccessSysfs": "false",
                            "Execution": {
                                "IsolationMode": "GreengrassContainer",
                                "RunAs": {
                                    "Uid": "1",
                                    "Gid": "10"
                                }
                            }
                        }
                    }
                }
            ]
        }
    },
    "TestLoggerDefinition": {
        "Type": "AWS::Greengrass::LoggerDefinition",
        "Properties": {
            "Name": "DemoTestLoggerDefinition"
        }
    },
    "TestLoggerDefinitionVersion": {
        "Type": "AWS::Greengrass::LoggerDefinitionVersion",
        "Properties": {
            "LoggerDefinitionId": {
```

```
                    "Ref": "TestLoggerDefinition"
                },
                "Loggers": [
                    {
                        "Id": "TestLogger1",
                        "Type": "AWSCloudWatch",
                        "Component": "GreengrassSystem",
                        "Level": "INFO"
                    }
                ]
            }
        },
        "TestResourceDefinition": {
            "Type": "AWS::Greengrass::ResourceDefinition",
            "Properties": {
                "Name": "DemoTestResourceDefinition"
            }
        },
        "TestResourceDefinitionVersion": {
            "Type": "AWS::Greengrass::ResourceDefinitionVersion",
            "Properties": {
                "ResourceDefinitionId": {
                    "Ref": "TestResourceDefinition"
                },
                "Resources": [
                    {
                        "Id": "ResourceId1",
                        "Name": "LocalDeviceResource",
                        "ResourceDataContainer": {
                            "LocalDeviceResourceData": {
                                "SourcePath": "/dev/TestSourcePath1",
                                "GroupOwnerSetting": {
                                    "AutoAddGroupOwner": "false",
                                    "GroupOwner": "TestOwner"
                                }
                            }
                        }
                    },
                    {
                        "Id": "ResourceId2",
                        "Name": "LocalVolumeResourceData",
                        "ResourceDataContainer": {
                            "LocalVolumeResourceData": {
                                "SourcePath": "/dev/TestSourcePath2",
```

```
                            "DestinationPath": "/volumes/TestDestinationPath2",
                            "GroupOwnerSetting": {
                                "AutoAddGroupOwner": "false",
                                "GroupOwner": "TestOwner"
                            }
                        }
                    }
                ]
            }
        },
        "TestSubscriptionDefinition": {
            "Type": "AWS::Greengrass::SubscriptionDefinition",
            "Properties": {
                "Name": "DemoTestSubscriptionDefinition"
            }
        },
        "TestSubscriptionDefinitionVersion": {
            "Type": "AWS::Greengrass::SubscriptionDefinitionVersion",
            "Properties": {
                "SubscriptionDefinitionId": {
                    "Ref": "TestSubscriptionDefinition"
                },
                "Subscriptions": [
                    {
                        "Id": "TestSubscription1",
                        "Source": {
                            "Fn::Join": [
                                ":",
                                [
                                    "arn:aws:iot",
                                    {
                                        "Ref": "AWS::Region"
                                    },
                                    {
                                        "Ref": "AWS::AccountId"
                                    },
                                    "thing/TestClientDevice1"
                                ]
                            ]
                        },
                        "Subject": "TestSubjectUpdated",
                        "Target": {
                            "Ref": "LambdaVersionArn"
```

```
                    }
                }
            ]
        }
    },
    "TestGroup": {
        "Type": "AWS::Greengrass::Group",
        "Properties": {
            "Name": "DemoTestGroupNewName",
            "RoleArn": {
                "Fn::Join": [
                    ":",
                    [
                        "arn:aws:iam:",
                        {
                            "Ref": "AWS::AccountId"
                        },
                        "role/TestUser"
                    ]
                ]
            },
            "InitialVersion": {
                "CoreDefinitionVersionArn": {
                    "Ref": "TestCoreDefinitionVersion"
                },
                "DeviceDefinitionVersionArn": {
                    "Ref": "TestDeviceDefinitionVersion"
                },
                "FunctionDefinitionVersionArn": {
                    "Ref": "TestFunctionDefinitionVersion"
                },
                "SubscriptionDefinitionVersionArn": {
                    "Ref": "TestSubscriptionDefinitionVersion"
                },
                "LoggerDefinitionVersionArn": {
                    "Ref": "TestLoggerDefinitionVersion"
                },
                "ResourceDefinitionVersionArn": {
                    "Ref": "TestResourceDefinitionVersion"
                }
            },
            "Tags": {
                "KeyName0": "value",
                "KeyName1": "value",
```

```
                    "KeyName2": "value"
                }
            }
        }
    },
    "Outputs": {
        "CommandToDeployGroup": {
            "Value": {
                "Fn::Join": [
                    " ",
                    [
                        "groupVersion=$(cut -d'/' -f6 <<<",
                        {
                            "Fn::GetAtt": [
                                "TestGroup",
                                "LatestVersionArn"
                            ]
                        },
                        ");",
                        "aws --region",
                        {
                            "Ref": "AWS::Region"
                        },
                        "greengrass create-deployment --group-id",
                        {
                            "Ref": "TestGroup"
                        },
                        "--deployment-type NewDeployment --group-version-id",
                        "$groupVersion"
                    ]
                ]
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: >-
  AWS IoT Greengrass example template that creates a group version with a core,
  device, function, logger, subscription, and resources.
Parameters:
```

```
  CoreCertificateArn:
    Type: String
  DeviceCertificateArn:
    Type: String
  LambdaVersionArn:
    Type: String
Resources:
  TestCore1:
    Type: 'AWS::IoT::Thing'
    Properties:
      ThingName: TestCore1
  TestCoreDefinition:
    Type: 'AWS::Greengrass::CoreDefinition'
    Properties:
      Name: DemoTestCoreDefinition
  TestCoreDefinitionVersion:
    Type: 'AWS::Greengrass::CoreDefinitionVersion'
    Properties:
      CoreDefinitionId: !Ref TestCoreDefinition
      Cores:
        - Id: TestCore1
          CertificateArn: !Ref CoreCertificateArn
          SyncShadow: 'false'
          ThingArn: !Join
            - ':'
            - - 'arn:aws:iot'
              - !Ref 'AWS::Region'
              - !Ref 'AWS::AccountId'
              - thing/TestCore1
  TestClientDevice1:
    Type: 'AWS::IoT::Thing'
    Properties:
      ThingName: TestClientDevice1
  TestDeviceDefinition:
    Type: 'AWS::Greengrass::DeviceDefinition'
    Properties:
      Name: DemoTestDeviceDefinition
  TestDeviceDefinitionVersion:
    Type: 'AWS::Greengrass::DeviceDefinitionVersion'
    Properties:
      DeviceDefinitionId: !GetAtt
        - TestDeviceDefinition
        - Id
      Devices:
```

```
      - Id: TestClientDevice1
        CertificateArn: !Ref DeviceCertificateArn
        SyncShadow: 'true'
        ThingArn: !Join
          - ':'
          - - 'arn:aws:iot'
            - !Ref 'AWS::Region'
            - !Ref 'AWS::AccountId'
            - thing/TestClientDevice1
TestFunctionDefinition:
  Type: 'AWS::Greengrass::FunctionDefinition'
  Properties:
    Name: DemoTestFunctionDefinition
TestFunctionDefinitionVersion:
  Type: 'AWS::Greengrass::FunctionDefinitionVersion'
  Properties:
    FunctionDefinitionId: !GetAtt
      - TestFunctionDefinition
      - Id
    DefaultConfig:
      Execution:
        IsolationMode: GreengrassContainer
    Functions:
      - Id: TestLambda1
        FunctionArn: !Ref LambdaVersionArn
        FunctionConfiguration:
          Pinned: 'true'
          Executable: run.exe
          ExecArgs: argument1
          MemorySize: '512'
          Timeout: '2000'
          EncodingType: binary
          Environment:
            Variables:
              variable1: value1
            ResourceAccessPolicies:
              - ResourceId: ResourceId1
                Permission: ro
              - ResourceId: ResourceId2
                Permission: rw
            AccessSysfs: 'false'
            Execution:
              IsolationMode: GreengrassContainer
              RunAs:
```

```
                Uid: '1'
                Gid: '10'
  TestLoggerDefinition:
    Type: 'AWS::Greengrass::LoggerDefinition'
    Properties:
      Name: DemoTestLoggerDefinition
  TestLoggerDefinitionVersion:
    Type: 'AWS::Greengrass::LoggerDefinitionVersion'
    Properties:
      LoggerDefinitionId: !Ref TestLoggerDefinition
      Loggers:
        - Id: TestLogger1
          Type: AWSCloudWatch
          Component: GreengrassSystem
          Level: INFO
  TestResourceDefinition:
    Type: 'AWS::Greengrass::ResourceDefinition'
    Properties:
      Name: DemoTestResourceDefinition
  TestResourceDefinitionVersion:
    Type: 'AWS::Greengrass::ResourceDefinitionVersion'
    Properties:
      ResourceDefinitionId: !Ref TestResourceDefinition
      Resources:
        - Id: ResourceId1
          Name: LocalDeviceResource
          ResourceDataContainer:
            LocalDeviceResourceData:
              SourcePath: /dev/TestSourcePath1
              GroupOwnerSetting:
                AutoAddGroupOwner: 'false'
                GroupOwner: TestOwner
        - Id: ResourceId2
          Name: LocalVolumeResourceData
          ResourceDataContainer:
            LocalVolumeResourceData:
              SourcePath: /dev/TestSourcePath2
              DestinationPath: /volumes/TestDestinationPath2
              GroupOwnerSetting:
                AutoAddGroupOwner: 'false'
                GroupOwner: TestOwner
  TestSubscriptionDefinition:
    Type: 'AWS::Greengrass::SubscriptionDefinition'
    Properties:
```

```
      Name: DemoTestSubscriptionDefinition
  TestSubscriptionDefinitionVersion:
    Type: 'AWS::Greengrass::SubscriptionDefinitionVersion'
    Properties:
      SubscriptionDefinitionId: !Ref TestSubscriptionDefinition
      Subscriptions:
        - Id: TestSubscription1
          Source: !Join
            - ':'
            - - 'arn:aws:iot'
              - !Ref 'AWS::Region'
              - !Ref 'AWS::AccountId'
              - thing/TestClientDevice1
          Subject: TestSubjectUpdated
          Target: !Ref LambdaVersionArn
  TestGroup:
    Type: 'AWS::Greengrass::Group'
    Properties:
      Name: DemoTestGroupNewName
      RoleArn: !Join
        - ':'
        - - 'arn:aws:iam:'
          - !Ref 'AWS::AccountId'
          - role/TestUser
      InitialVersion:
        CoreDefinitionVersionArn: !Ref TestCoreDefinitionVersion
        DeviceDefinitionVersionArn: !Ref TestDeviceDefinitionVersion
        FunctionDefinitionVersionArn: !Ref TestFunctionDefinitionVersion
        SubscriptionDefinitionVersionArn: !Ref TestSubscriptionDefinitionVersion
        LoggerDefinitionVersionArn: !Ref TestLoggerDefinitionVersion
        ResourceDefinitionVersionArn: !Ref TestResourceDefinitionVersion
      Tags:
        KeyName0: value
        KeyName1: value
        KeyName2: value
Outputs:
  CommandToDeployGroup:
    Value: !Join
      - ' '
      - - groupVersion=$(cut -d'/' -f6 <<<
        - !GetAtt
          - TestGroup
          - LatestVersionArn
        - );
```

```
        - aws --region
        - !Ref 'AWS::Region'
        - greengrass create-deployment --group-id
        - !Ref TestGroup
        - '--deployment-type NewDeployment --group-version-id'
        - $groupVersion
```

Didukung Wilayah AWS s

Saat ini, Anda dapat membuat dan mengelola AWS IoT Greengrass sumber daya hanya dalam berikut Wilayah AWSs:

- US East (Ohio)
- AS Timur (Virginia Utara)
- US West (Oregon)
- Asia Pasifik (Mumbai)
- Asia Pasifik (Seoul)
- Asia Pasifik (Singapura)
- Asia Pasifik (Sydney)
- Asia Pasifik (Tokyo)
- China (Beijing)
- Eropa (Frankfurt)
- Eropa (Irlandia)
- Eropa (London)
- AWS GovCloud (AS-Barat)

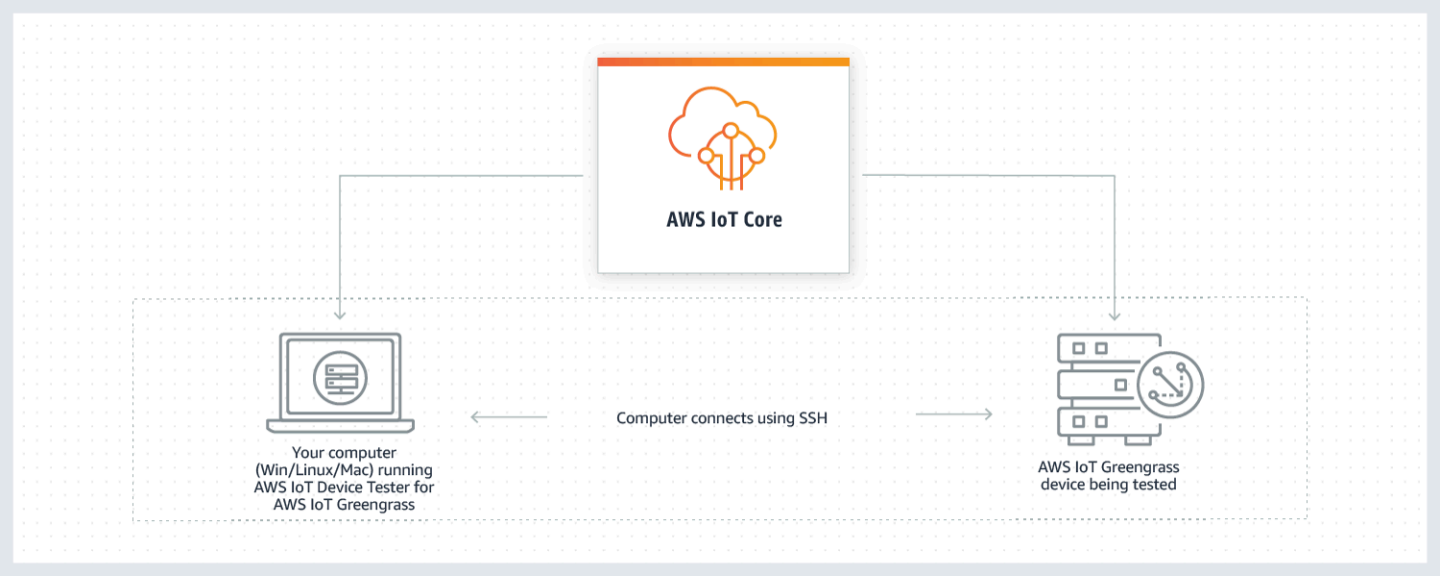
Menggunakan AWS IoT Penguji Perangkat untuk AWS IoT Greengrass V1

AWS IoT Penguji Perangkat (IDT) adalah kerangka pengujian unduh yang memungkinkan Anda memvalidasi perangkat IoT. KarenaAWS IoT Greengrass Version 1telah dipindahkan kemodus pemeliharaan, IDT untukAWS IoT Greengrass V1tidak lagi menghasilkan laporan kualifikasi yang ditandatangani. Anda tidak akan lagi dapat memenuhi syarat baruAWS IoT Greengrass V1perangkat untuk daftar diAWS PartnerKatalog perangkatmelaluiAWSProgram Kualifikasi Perangkat. Namun, Anda dapat terus menggunakan IDT untukAWS IoT Greengrass V1untuk menguji perangkat Greengrass V1 Anda. Kami menyarankan agar Anda menggunakanIDT untukAWS IoT Greengrass V2untuk memenuhi syarat dan daftar perangkat Greengrass diAWS PartnerKatalog perangkat.

IDT untuk AWS IoT Greengrass berjalan di komputer host Anda (Windows, macOS, atau Linux) yang terhubung ke perangkat yang akan diuji. IDT menjalankan tes dan mengelompokkan hasil. IDT juga menyediakan antarmuka baris perintah untuk mengelola proses pengujian.

AWS IoT Greengrass suite kualifikasi

Gunakan IDT untuk AWS IoT Greengrass untuk memverifikasi bahwa AWS IoT Greengrass perangkat lunak Core berjalan pada perangkat keras Anda dan dapat berkomunikasi dengan AWS Cloud. Hal ini juga melakukan end-to-end pengujian denganAWS IoT Core. Sebagai contoh, perangkat ini memverifikasi bahwa perangkat Anda dapat mengirim dan menerima pesan MQTT dan memprosesnya dengan benar.



AWS IoT Penguji Perangkat untuk AWS IoT Greengrass mengatur pengujian menggunakan konsep-konsep suite pengujian dan grup pengujian.

- Rangkaian percobaan adalah kumpulan grup pengujian yang digunakan untuk memverifikasi bahwa perangkat bekerja dengan versi tertentu AWS IoT Greengrass.
- Sebuah grup pengujian adalah seperangkat pengujian individu yang terkait dengan fitur tertentu, seperti deployment grup Greengrass dan pesan MQTT.

Untuk informasi selengkapnya, lihat Gunakan IDT untuk menjalankan AWS IoT Greengrass suite kualifikasi.

Rangkaian pengujian khusus

Memulai di IDT v4.0.0, IDT untuk AWS IoT Greengrass menggabungkan pengaturan konfigurasi terstandar dan format hasil dengan lingkungan rangkaian percobaan yang memungkinkan Anda untuk mengembangkan rangkaian pengujian khusus untuk perangkat dan perangkat lunak Anda. Anda dapat menambahkan pengujian khusus untuk validasi internal Anda sendiri atau memberikannya kepada pelanggan Anda untuk verifikasi perangkat.

Cara penyusun tes mengonfigurasi rangkiaian tes kustom akan menentukan pengaturan konfigurasi yang diperlukan untuk menjalankan rangkaian tes kustom. Untuk informasi selengkapnya, lihat Gunakan IDT untuk mengembangkan dan menjalankan rangkaian pengujian Anda sendiri.

Versi yang didukung AWS IoT Penguji Perangkat untuk AWS IoT Greengrass V1

Karena AWS IoT Greengrass Version 1 telah dipindahkan ke mode pemeliharaan, IDT AWS IoT Greengrass V1 tidak lagi menghasilkan laporan kualifikasi yang ditandatangani. Kami menyarankan Anda menggunakan IDT untuk AWS IoT Greengrass V2.

Untuk informasi tentang IDT untuk AWS IoT Greengrass V2, lihat Menggunakan AWS IoT Penguji Perangkat untuk AWS IoT Greengrass V2 di AWS IoT Greengrass V2 Panduan Developer.

> **Note**
>
> Anda menerima notifikasi ketika Anda memulai uji coba jika IDT untuk AWS IoT Greengrass tidak kompatibel dengan versi yang AWS IoT Greengrass Anda gunakan.

Dengan mengunduh perangkat lunak, Anda menyetujui AWS IoT Perjanjian Lisensi Penguji Perangkat.

Versi IDT yang tidak didukung untuk AWS IoT Greengrass

Topik ini mencantumkan versi IDT yang tidak didukung untuk AWS IoT Greengrass. Versi yang tidak didukung tidak menerima perbaikan bug atau pembaruan. Untuk informasi selengkapnya, lihat the section called “Support kebijakan untuk AWS IoT Penguji Perangkat untuk AWS IoT Greengrass V1”.

IDT v4.4.1 untuk AWS IoT Greengrass versi v1.11.6, v1.10.5

Catatan rilis:

- Memungkinkan Anda memvalidasi dan memenuhi syarat perangkat yang menjalankan perangkat lunak AWS IoT Greengrass inti v1.11.6 dan v1.10.5.
- Berisi perbaikan bug minor.

Versi rangkaian tes:

`GGQ_1.3.1`

- Dirilis 2021.12.20

IDT v4.1.0 untuk AWS IoT Greengrass versi v1.11.4, v1.10.4

Catatan rilis:

- Memungkinkan Anda memvalidasi dan memenuhi syarat perangkat yang menjalankan perangkat lunak AWS IoT Greengrass inti v1.11.4 dan v1.10.4.
- Memperbaiki masalah yang menyebabkan log yang ditampilkan selama uji coba menggunakan tag berlebihan.

Versi suite pengujian:

`GGQ_1.3.0`

- Dirilis 2021.06.23
- Menambahkan retries untuk panggilan API untuk Lambda, IAM, dan AWS STS untuk meningkatkan penanganan untuk masalah throttling atau server.
- Menambahkan support untuk Python 3.8 untuk kasus uji ML dan Docker.

IDT v4.0.2 untuk AWS IoT Greengrass versi v1.11.1, v1.11.0, v1.10.3

Catatan rilis:

- Memperbaiki masalah yang menyebabkan IDT untuk menutupi kesalahan Hardware Security Integration (HSI).
- Memungkinkan Anda untuk mengembangkan dan menjalankan rangkaian pengujian khusus Anda menggunakan AWS IoT Penguji Perangkat untuk AWS IoT Greengrass. Untuk informasi selengkapnya, lihat Gunakan IDT untuk mengembangkan dan menjalankan rangkaian pengujian Anda sendiri.
- Menyediakan kode aplikasi IDT yang ditandatangani untuk macOS dan Windows. Di macOS, jika pesan peringatan keamanan ditampilkan, Anda mungkin harus memberikan pengecualian keamanan untuk IDT. Untuk informasi selengkapnya, lihat Pengecualian keamanan di macOS.

> **Note**
>
> AWS IoT Greengrass tidak menyediakan Dockerfile atau gambar Docker untuk versi 1.11.1 AWS IoT Greengrass perangkat lunak core. Untuk menguji kualifikasi Docker di perangkat, gunakan versi sebelumnya AWS IoT Greengrass perangkat lunak core.

IDT v3.2.0 untuk AWS IoT Greengrass versi v1.11.0, v1.10.1, v1.10.0

Catatan rilis:

- Secara default, IDT hanya menjalankan pengujian yang diperlukan untuk kualifikasi. Untuk memenuhi syarat untuk fitur tambahan, Anda dapat memodifikasi `device.json` file.
- Ditambahkan nomor port di `device.json` yang dapat Anda konfigurasikan untuk koneksi SSH.
- Docker hanya mendukung pengelola pengaliran dan machine learning (ML) tanpa kontainerisasi. Kontainer, Docker, dan Hardware Security Integration (HSI) tidak tersedia untuk perangkat Docker.
- Kami gabungkan `device-ml.json` dan `device-hsm.json` ke `device.json`.

IDT v3.1.3 untuk AWS IoT Greengrass versi: v1.10.x, v1.9.x, v1.8.x

Catatan rilis:

- Ditambahkan support untuk kualifikasi fitur ML untuk AWS IoT Greengrass v1.10.x dan v1.9.x. Anda sekarang dapat menggunakan IDT untuk memvalidasi bahwa perangkat Anda dapat melakukan infrensi ML lokal dengan model yang disimpan dan dilatih di cloud.

- Ditambahkan `--stop-on-first-failure` untuk `run-suite` perintah. Anda dapat menggunakan opsi ini untuk mengkonfigurasi IDT untuk berhenti berjalan pada kegagalan pertama. Kami merekomendasikan menggunakan opsi ini selama tahap debugging pada tingkat grup uji.
- Ditambahkan cek drift clock untuk pengujian MQTT untuk memastikan bahwa perangkat yang diuji menggunakan waktu sistem yang benar. Waktu yang digunakan harus berada dalam rentang waktu yang dapat diterima.
- Ditambahkan `--update-idt` untuk `run-suite` perintah. Anda dapat menggunakan opsi ini untuk mengatur respons untuk prompt untuk update IDT.
- Ditambahkan `--update-managed-policy` untuk `run-suite` perintah. Anda dapat menggunakan opsi ini untuk mengatur respons untuk prompt untuk update kebijakan terkelola.
- Ditambahkan perbaikan bug untuk pembaruan otomatis dari versi rangkaian pengujian IDT. Perbaikan memastikan bahwa IDT dapat menjalankan suite pengujian terbaru yang tersedia untuk versi AWS IoT Greengrass Anda.

IDT v3.0.1 untuk AWS IoT Greengrass

Catatan rilis:

- Ditambahkan support untuk AWS IoT Greengrass v1.10.1.
- Pembaruan otomatis IDTuji suite versi. IDT dapat mengunduh suite pengujian terbaru yang tersedia untuk versi AWS IoT Greengrass Anda. Dengan fitur ini:
  - Suite pengujian diversi menggunakan *`major`*`.`*`minor`*`.`*`patch`* format. Versi suite pengujian awal adalah `GGQ_1.0.0`.
  - Anda dapat mengunduh suite pengujian baru interaktif di antarmuka baris perintah atau mengatur `upgrade-test-suite` bendera ketika Anda mulai IDT.

  Untuk informasi selengkapnya, lihat the section called “Versi rangkaian pengujian”.
- Ditambahkan `list-supported-products`. Anda dapat menggunakan perintah ini untuk daftar AWS IoT Greengrass dan pengujian suite versi yang didukung oleh versi diinstal IDT.
- Ditambahkan `list-test-cases`. Anda dapat menggunakan perintah ini untuk daftar kasus pengujian yang tersedia dalam grup pengujian.
- Ditambahkan `test-id` untuk `run-suite` perintah. Anda dapat menggunakan opsi ini untuk menjalankan kasus pengujian individu dalam grup pengujian.

IDT v2.3.0 untuk AWS IoT Greengrass v1.10, v1.9.x, dan v1.8.x

Saat menguji pada perangkat fisik, AWS IoT Greengrass v1.10, v1.9.x, dan v1.8.x didukung.

Ketika pengujian dalam kontainer Docker, AWS IoT Greengrass v1.10 dan v1.9.x didukung.

Catatan rilis:

- Ditambahkan support untuk the section called “Jalankan AWS IoT Greengrass di kontainer Docker”. Anda sekarang dapat menggunakan IDT untuk memenuhi syarat dan memvalidasi bahwa perangkat Anda dapat menjalankan AWS IoT Greengrass dalam kontainer Docker.
- Ditambahkan sebuah AWS kebijakan terkelola (`AWSIoTDeviceTesterForGreengrassFullAccess`) yang mendefinisikan izin yang diperlukan untuk menjalankan AWS IoT Penguji Perangkat. Jika rilis baru memerlukan izin tambahan, AWS menambahkannya ke kebijakan terkelola ini sehingga Anda tidak perlu update izin IAM.
- Pemeriksaan diperkenalkan untuk memvalidasi bahwa lingkungan Anda (misalnya, konektivitas perangkat dan konektivitas internet) diatur dengan benar sebelum Anda menjalankan kasus pengujian.
- Peningkatan checker dependensi Greengrass di IDT untuk membuatnya lebih fleksibel saat memeriksa libc pada perangkat.

IDT v2.2.0 untuk AWS IoT Greengrass v1.10, v1.9.x, dan v1.8.x

Catatan rilis:

- Ditambahkan support untuk AWS IoT Greengrass v1.10.
- Ditambahkan support untuk konektor deployment aplikasi Greengrass Docker.
- Ditambahkan support untuk AWS IoT Greengrass pengelola pengaliran.
- Ditambahkan support untuk AWS IoT Greengrass di Wilayah China (Beijing).

IDT v2.1.0 untuk AWS IoT Greengrass v1.9.x, v1.8.x, dan v1.7.x

Catatan rilis:

- Ditambahkan support untuk AWS IoT Greengrass v1.9.4.
- Ditambahkan support untuk perangkat Linux-ARMv6l.

IDT v2.0.0 untuk AWS IoT Greengrass v1.9.3, v1.9.2, v.1.9.1, v1.9.0, v1.8.4, v1.8.3, dan v1.8.2

Catatan rilis:

- Dihapus dependensi pada Python untuk perangkat yang diuji.
- Waktu eksekusi pengujian suite dikurangi lebih dari 50 persen, yang membuat proses kualifikasi lebih cepat.
- Ukuran yang dapat dieksekusi dikurangi lebih dari 50 persen, yang membuat pengunduhan dan pemasangan lebih cepat.
- Peningkatan support pengali waktu habis untuk semua kasus pengujian.
- Pesan pasca-diagnostik yang ditingkatkan untuk memecahkan masalah kesalahan dengan lebih cepat.
- Update templat kebijakan izin yang diperlukan untuk menjalankan IDT.
- Ditambahkan support untuk AWS IoT Greengrass v1.9.3.

IDT v1.3.3 untuk AWS IoT Greengrass v1.9.2, v1.9.1, v1.9.0, v1.8.3, dan v1.8.2

Catatan rilis:

- Ditambahkan support untuk Greengrass v1.9.2 dan v1.8.3.
- Menambahkan dukungan untuk Greengrass OpenWrt.
- Ditambahkan nama pengguna SSH dan perangkat kata sandi masuk.
- Menambahkan perbaikan bug uji asli untuk platform OpenWrt -ARMv7L.

IDT v1.2 untuk v1.8.1 AWS IoT Greengrass

Catatan rilis:

- Ditambahkan pengali waktu habis yang dapat dikonfigurasi untuk mengatasi dan memecahkan masalah waktu habis (sebagai contoh, koneksi bandwidth rendah).

IDT v1.1 untuk v1.8.0 AWS IoT Greengrass

Catatan rilis:

- Ditambahkan support untuk AWS IoT Greengrass Hardware Security Integration (HSI).

- Ditambahkan support untuk AWS IoT Greengrass kontainer dan tanpa kontainer.
- Ditambahkan otomatis AWS IoT Greengrass pembuatan peran layanan.
- Peningkatan pembersihan sumber daya pengujian.
- Ditambahkan laporan ringkasan eksekusi pengujian.

IDT v1.1 untuk v1.7.1 AWS IoT Greengrass

Catatan rilis:

- Ditambahkan support untuk AWS IoT Greengrass Hardware Security Integration (HSI).
- Ditambahkan support untuk AWS IoT Greengrass kontainer dan tanpa kontainer.
- Ditambahkan otomatis AWS IoT Greengrass pembuatan peran layanan.
- Peningkatan pembersihan sumber daya pengujian.
- Ditambahkan laporan ringkasan eksekusi pengujian.

IDT v1.0 untuk v1.6.1 AWS IoT Greengrass

Catatan rilis:

- Ditambahkan perbaikan bug pengujian OTA untuk kompatibilitas versi AWS IoT Greengrass masa depan.

> **Note**
>
> Jika Anda menggunakan IDT v1.0 untuk AWS IoT Greengrass v1.6.1, Anda harus membuat peran layanan Greengrass. Dalam versi yang lebih baru, IDT membuat peran layanan untuk Anda.

Gunakan IDT untuk menjalankan AWS IoT Greengrass suite kualifikasi

Anda dapat menggunakan AWS IoT Tester Perangkat (IDT) untuk AWS IoT Greengrass untuk memverifikasi bahwa AWS IoT Greengrass perangkat lunak Core berjalan pada perangkat keras Anda dan dapat berkomunikasi dengan AWS Cloud. Itu juga melakukan end-to-end tes denganAWS

IoT Core. Sebagai contoh, perangkat ini memverifikasi bahwa perangkat Anda dapat mengirim dan menerima pesan MQTT dan memprosesnya dengan benar.

KarenaAWS IoT Greengrass Version 1telah dipindahkan kemodus pemeliharaan, IDT untukAWS IoT Greengrass V1tidak lagi menghasilkan laporan kualifikasi yang ditandatangani. Jika Anda ingin menambahkan perangkat keras Anda ke Katalog Perangkat AWS Partner, jalankan AWS IoT Greengrass V2 untuk menghasilkan laporan pengujian yang dapat Anda kirimkan ke AWS IoT. Untuk informasi selengkapnya, lihatAWSProgram Kualifikasi PerangkatdanVersi IDT yang didukung untukAWS IoT Greengrass V2.

Selain perangkat pengetesan, IDT untuk AWS IoT Greengrass membuat sumber daya (sebagai contoh, AWS IoT hal-hal, AWS IoT Greengrass grup, fungsi Lambda, dan sebagainya) di Anda Akun AWS untuk memfasilitasi proses kualifikasi.

Untuk membuat sumber daya ini, IDT untuk AWS IoT Greengrass menggunakan AWS kredensial yang dikonfigurasi dalam file `config.json` untuk melakukan panggilan API atas nama Anda. Sumber daya ini ditetapkan pada berbagai waktu selama tes.

Ketika Anda menggunakan IDT AWS IoT Greengrass untuk menjalankan AWS IoT Greengrass suite kualifikasi, IDT melakukan langkah-langkah berikut:

1. Memuat dan memvalidasi perangkat dan konfigurasi kredensial Anda.
2. Melakukan tes yang dipilih dengan sumber daya lokal dan cloud yang diperlukan.
3. Membersihkan sumber daya lokal dan cloud.
4. Menghasilkan laporan tes yang menunjukkan jika perangkat Anda lulus tes yang diperlukan untuk kualifikasi.

Versi rangkaian pengujian

IDT untuk AWS IoT Greengrass mengatur tes ke tes suite dan grup tes.

- Rangkaian percobaan adalah kumpulan grup tes yang digunakan untuk memverifikasi bahwa perangkat bekerja dengan versi tertentu AWS IoT Greengrass.
- Sebuah grup tes adalah seperangkat tes individu yang terkait dengan fitur tertentu, seperti deployment grup Greengrass dan pesan MQTT.

Dimulai di IDT v3.0.0, test suite diversi menggunakan *`major.minor.patch`* format, sebagai contoh `GGQ_1.0.0`. Ketika Anda mengunduh IDT, paket termasuk versi test suite terbaru.

> **Important**
>
> IDT mendukung tiga versi rangkaian pengujian terbaru untuk kualifikasi perangkat. Untuk informasi selengkapnya, lihat the section called “Support kebijakan untuk AWS IoT Penguji Perangkat untuk AWS IoT Greengrass V1”.
>
> Anda dapat menjalankan `list-supported-products` untuk mencantumkan versi AWS IoT Greengrass dan test suite yang didukung oleh versi IDT Anda saat ini. Pengujian dari versi rangkaian pengujian yang tidak didukung, tidak valid untuk kualifikasi perangkat. IDT tidak mencetak laporan kualifikasi untuk versi yang tidak didukung.

Update untuk pengaturan konfigurasi IDT

Tes baru mungkin memperkenalkan pengaturan konfigurasi IDT baru.

- Jika pengaturan opsional, IDT terus menjalankan tes.
- Jika pengaturan diperlukan, IDT memberitahu Anda dan berhenti berjalan. Setelah Anda mengkonfigurasi pengaturan, restart uji coba.

  Pengaturan konfigurasi terletak di *<device-tester-extract-location>*`/configs` folder. Untuk informasi selengkapnya, lihat the section called “Konfigurasikan pengaturan IDT”.

Jika versi suite tes di-update menambahkan pengaturan konfigurasi, IDT membuat salinan file konfigurasi asli di *<device-tester-extract-location>*`/configs`.

Deskripsi grup tes

IDT v2.0.0 and later

Grup Uji Wajib untuk Kualifikasi Inti

Kelompok tes ini diwajibkan untuk memenuhi syarat AWS IoT Greengrass perangkat untuk perangkat AWS Partner Katalog Perangkat.

AWS IoT Greengrass Dependensi Core

Memvalidasi bahwa perangkat Anda memenuhi semua persyaratan perangkat lunak dan perangkat keras untuk AWS IoT Greengrass perangkat lunak Core.

Tes `Software Packages Dependencies` kasus dalam grup tes ini tidak berlaku ketika pengetesan di dalam kontainer Docker.

Deployment

Memvalidasi bahwa fungsi Lambda dapat digunakan pada perangkat Anda.

MQTT

MemverifikasiAWS IoT Greengrassfungsi router pesan dengan memeriksa komunikasi lokal antara Greengrass core dan Perangkat klien, yang Perangkat IoT lokal.

Lewat udara (OTA)

Memvalidasi bahwa perangkat Anda dapat berhasil melakukan update OTA AWS IoT Greengrass perangkat lunak Core.

Grup tes ini tidak berlaku ketika pengetesan di kontainer Docker.

Versi

Memeriksa bahwa versi AWS IoT Greengrass yang disediakan kompatibel dengan AWS IoT versi Tester Perangkat yang Anda gunakan.

Grup Pengujian Opsional

Grup uji ini bersifat opsional. Jika Anda memilih untuk memenuhi syarat untuk pengujian opsional, perangkat Anda terdaftar dengan kemampuan tambahan di Katalog Perangkat AWS Partner ini.

Dependensi Kontainer

Memvalidasi bahwa perangkat memenuhi semua persyaratan perangkat lunak dan perangkat keras untuk menjalankan fungsi Lambda dalam mode kontainer pada core Greengrass.

Grup tes ini tidak berlaku ketika pengetesan di kontainer Docker.

Kontainer Deployment

Memvalidasi bahwa fungsi Lambda dapat digunakan pada perangkat dan berjalan dalam mode kontainer pada Greengrass core.

Grup tes ini tidak berlaku ketika pengetesan di kontainer Docker.

Docker Dependensi (Didukung untuk IDT v2.2.0 dan yang lebih baru)

Memvalidasi bahwa perangkat memenuhi semua dependensi teknis yang diperlukan untuk menggunakan konektor deployment aplikasi Greengrass Docker untuk menjalankan kontainer

Grup tes ini tidak berlaku ketika pengetesan di kontainer Docker.

Integrasi Keamanan Perangkat Keras (HSI)

Memverifikasi bahwa perpustakaan bersama HSI yang disediakan dapat antarmuka dengan modul keamanan perangkat keras (HSM) dan mengimplementasikan PKCS #11 API yang diperlukan dengan benar. HSM dan perpustakaan bersama harus dapat menandatangani CSR, melakukan operasi TLS, dan memberikan panjang kunci yang benar dan algoritma kunci publik.

Stream Manager Dependensi (Didukung untuk IDT v2.2.0 dan yang lebih baru)

Memvalidasi bahwa perangkat memenuhi semua dependensi teknis yang diperlukan untuk menjalankan AWS IoT Greengrass pengelola aliran.

Dependensi Machine Learning (Didukung untuk IDT v3.1.0 dan yang lebih baru)

Memvalidasi bahwa perangkat memenuhi semua dependensi teknis yang diperlukan untuk melakukan inferensi ML secara lokal.

Pengujian Inferensi Machine Learning (Didukung untuk IDT v3.1.0 dan versi lebih baru)

Memvalidasi bahwa inferensi ML dapat dilakukan pada perangkat yang diberikan dites. Untuk informasi selengkapnya, lihat the section called “Opsional: Mengonfigurasi perangkat Anda untuk kualifikasi ML-nya”.

Tes Kontainer Inferensi Machine Learning (Didukung untuk IDT v3.1.0 dan yang lebih baru)

Memvalidasi bahwa inferensi ML dapat dilakukan pada perangkat tertentu yang sedang dites dan dijalankan dalam mode kontainer pada Greengrass core. Untuk informasi selengkapnya, lihat the section called “Opsional: Mengonfigurasi perangkat Anda untuk kualifikasi ML-nya”.

IDT v1.3.3 and earlier

Grup Uji Wajib untuk Kualifikasi Inti

Tes ini diperlukan untuk memenuhi syarat AWS IoT Greengrass perangkat untuk perangkat AWS Partner Katalog Perangkat.

AWS IoT Greengrass Dependensi Core

Memvalidasi bahwa perangkat Anda memenuhi semua persyaratan perangkat lunak dan perangkat keras untuk AWS IoT Greengrass perangkat lunak Core.

Kombinasi (Interaksi Keamanan Perangkat)

Memverifikasi fungsionalitas certificate manager perangkat dan deteksi IP pada perangkat Greengrass core dengan mengubah informasi konektivitas pada grup Greengrass di cloud. Grup tes memutar AWS IoT Greengrass sertifikat server dan memverifikasi bahwa AWS IoT Greengrass mengizinkan koneksi.

Deployment (Diperlukan untuk IDT v1.2 dan sebelumnya)

Memvalidasi bahwa fungsi Lambda dapat digunakan pada perangkat Anda.

Device Certificate Manager (DCM)

Verifikasi bahwa perangkat AWS IoT Greengrass certificate manager dapat menghasilkan sertifikat server pada startup dan memutar sertifikat jika mereka mendekati kedaluwarsa.

Deteksi IP (IPD)

Memverifikasi bahwa informasi konektivitas core di-update ketika ada perubahan alamat IP dalam perangkat Greengrass core. Untuk informasi selengkapnya, lihat Aktifkan deteksi IP otomatis.

Mencatat

Verifikasi bahwa AWS IoT Greengrass layanan pencatatan dapat menulis ke berkas log menggunakan fungsi pengguna Lambda ditulis dengan Python.

MQTT

Memverifikasi AWS IoT Greengrass fungsionalitas router pesan dengan mengirim pesan pada topik yang diarahkan ke dua fungsi Lambda.

Asli

Memverifikasi bahwa AWS IoT Greengrass dapat menjalankan fungsi Lambda asli (dikompilasi).

Lewat udara (OTA)

Memvalidasi bahwa perangkat Anda dapat berhasil melakukan pembaruan OTA AWS IoT Greengrass perangkat lunak Core.

penetrasi

Memvalidasi bahwa AWS IoT Greengrass perangkat lunak core gagal dimulai jika perlindungan hard link/soft link dan seccomp tidak diaktifkan. Hal ini juga digunakan untuk memverifikasi fitur terkait keamanan lainnya.

Bayangan

Memverifikasi fungsi bayangan lokal dan bayangan cloud-syncing.

Spooler

Memvalidasi bahwa pesan MQTT antri dengan konfigurasi spooler default.

Layanan Bursa Token (TES)

Memverifikasi bahwa AWS IoT Greengrass dapat bertukar sertifikat Core untuk valid AWS kredenensial.

Versi

Memeriksa bahwa versi AWS IoT Greengrass yang disediakan kompatibel dengan AWS IoT versi Tester Perangkat yang Anda gunakan.

Grup Pengujian Opsional

Tes ini opsional. Jika Anda memilih untuk memenuhi syarat untuk pengujian opsional, perangkat Anda terdaftar dengan kemampuan tambahan di Katalog Perangkat AWS Partner ini.

Dependensi Kontainer

Cek bahwa perangkat memenuhi semua dependensi yang diperlukan untuk menjalankan fungsi Lambda dalam mode kontainer.

Integrasi Keamanan Perangkat Keras (HSI)

Memverifikasi bahwa perpustakaan bersama HSI yang disediakan dapat antarmuka dengan modul keamanan perangkat keras (HSM) dan mengimplementasikan PKCS #11 API yang diperlukan dengan benar. HSM dan perpustakaan bersama harus dapat menandatangani CSR, melakukan operasi TLS, dan memberikan panjang kunci yang benar dan algoritma kunci publik.

Akses Sumber Daya Lokal

Memverifikasi akses sumber daya lokal (LRA) fitur AWS IoT Greengrass dengan menyediakan akses ke file lokal dan direktori yang dimiliki oleh berbagai pengguna dan grup Linux untuk fungsi Lambda yang terkontainerisasi melalui AWS IoT Greengrass API LRA. Fungsi Lambda harus diizinkan atau ditolak akses ke sumber daya lokal berdasarkan konfigurasi akses sumber daya lokal.

Jaringan

Memverifikasi bahwa koneksi soket dapat dibentuk dari fungsi Lambda. Koneksi soket ini harus diizinkan atau ditolak berdasarkan konfigurasi Greengrass core.

Prasyarat untuk menjalankan rangkaian kualifikasi AWS IoT Greengrass

Bagian ini menjelaskan prasyarat untuk menggunakan AWS IoT Tester Perangkat (IDT) untuk AWS IoT Greengrass untuk menjalankan AWS IoT Greengrass suite kualifikasi.

Unduh versi terbaru AWS IoT Tester Perangkat AWS IoT Greengrass

Unduh versi terbaru IDT dan mengekstraksi perangkat lunak ke lokasi pada sistem file Anda di mana Anda telah membaca dan menulis izin.

> **Note**
>
> IDT tidak mendukung yang sedang dijalankan oleh beberapa pengguna dari lokasi bersama, seperti direktori NFS atau folder bersama jaringan Windows. Kami sarankan Anda mengekstraksi paket IDT ke drive lokal dan menjalankan biner IDT pada workstation lokal Anda.
> Windows memiliki batasan panjang jalur 260 karakter. Jika Anda menggunakan Windows, mengekstraksi IDT ke direktori root seperti `C:\` atau `D:\` untuk menjaga jalur Anda di bawah batas 260 karakter.

Buat dan konfigurasi pengguna Akun AWS

Sebelum Anda dapat menggunakan IDT untuk AWS IoT Greengrass, Anda harus melakukan langkah-langkah berikut:

1. Buat sebuahAkun AWS. Jika Anda sudah memiliki Akun AWS, lewati ke langkah 2.
2. Konfigurasikan izin untuk IDT.

Izin akun ini memungkinkan IDT untuk mengakses AWS layanan dan membuat AWS sumber daya, seperti AWS IoT hal, grup Greengrass, dan fungsi Lambda, atas nama Anda.

Untuk membuat sumber daya ini, IDT untuk AWS IoT Greengrass menggunakan AWS kredensial yang dikonfigurasi dalam file `config.json` untuk melakukan panggilan API atas nama Anda. Sumber daya ini ditetapkan pada berbagai waktu selama tes.

> **Note**
>
> Meskipun sebagian besar tes memenuhi syarat untuk Amazon Web Services Tingkat Gratis, Anda harus menyediakan kartu kredit saat mendaftar Akun AWS. Untuk informasi lebih lanjut, lihat Mengapa saya memerlukan metode pembayaran jika akun saya dilindungi oleh Tingkat Gratis?.

Langkah 1: Buat Akun AWS

Pada langkah ini, buat dan konfigurasikan Akun AWS. Jika Anda sudah memiliki akun Akun AWS, lewati ke the section called “Langkah 2: Konfigurasi izin untuk IDT”.

Mendaftar Akun AWS

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar Akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.
2. Ikuti petunjuk secara online.

   Anda akan diminta untuk menerima panggilan telepon dan memasukkan kode verifikasi pada keypad telepon sebagai bagian dari prosedur pendaftaran.

Saat Anda mendaftar Akun AWS, Pengguna root akun AWS akan dibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya dalam akun. Sebagai praktik terbaik keamanan, tetapkan akses administratif ke pengguna administratif, dan hanya gunakan pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

AWS akan mengirimkan email konfirmasi kepada Anda setelah proses pendaftaran selesai. Anda dapat melihat aktivitas akun saat ini dan mengelola akun dengan mengunjungi https://aws.amazon.com/ dan memilih Akun Saya.

Membuat pengguna administratif

Setelah mendaftar Akun AWS, amankan Pengguna root akun AWS, aktifkan AWS IAM Identity Center, dan buat sebuah pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Mengamankan Pengguna root akun AWS Anda

1. Masuk ke AWS Management Console sebagai pemilik akun dengan memilih Pengguna root dan memasukkan alamat email Akun AWS Anda. Di halaman berikutnya, masukkan kata sandi Anda.

   Untuk bantuan masuk menggunakan pengguna root, lihat Masuk sebagai pengguna root dalam Panduan Pengguna AWS Sign-In.
2. Aktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda.

   Untuk petunjuknya, silakan lihat Mengaktifkan perangkat MFA virtual untuk pengguna root Akun AWS Anda (konsol) dalam Panduan Pengguna IAM.

Membuat pengguna administratif

1. Aktifkan Pusat Identitas IAM.

   Untuk mendapatkan petunjuk, silakan lihat Mengaktifkan AWS IAM Identity Center di Panduan Pengguna AWS IAM Identity Center.
2. Di Pusat Identitas IAM, berikan akses administratif ke sebuah pengguna administratif.

   Untuk mendapatkan tutorial tentang menggunakan Direktori Pusat Identitas IAM sebagai sumber identitas Anda, silakan lihat Mengonfigurasi akses pengguna dengan Direktori Pusat Identitas IAM default di Panduan Pengguna AWS IAM Identity Center.

Masuk sebagai pengguna administratif

- Untuk masuk dengan pengguna Pusat Identitas IAM, gunakan URL masuk yang dikirim ke alamat email Anda saat Anda membuat pengguna Pusat Identitas IAM.

  Untuk bantuan masuk menggunakan pengguna Pusat Identitas IAM, lihat Masuk ke portal akses AWS dalam Panduan Pengguna AWS Sign-In.

Langkah 2: Konfigurasi izin untuk IDT

Pada langkah ini, konfigurasi izin yang IDT untuk AWS IoT Greengrass menggunakan untuk menjalankan tes dan mengumpulkan data penggunaan IDT. Anda dapat menggunakan AWS Management Console atau AWS Command Line Interface (AWS CLI) untuk membuat kebijakan IAM dan pengguna tes untuk IDT, dan kemudian melampirkan kebijakan untuk pengguna. Jika Anda telah membuat pengguna percobaan untuk IDT, lewati ke the section called “Konfigurasikan perangkat Anda untuk menjalankan tes IDT” atau the section called “Opsional: Mengonfigurasi kontainer Docker”.

- Untuk Mengkonfigurasi Izin untuk IDT (Konsol)
- Untuk Mengkonfigurasi Izin untuk IDT () AWS CLI

Untuk mengonfigurasi izin untuk IDT (konsol)

Ikuti langkah berikut untuk menggunakan konsol untuk mengonfigurasi izin untuk IDT untuk AWS IoT Greengrass.

1. Masuklah ke konsol IAM.
2. Buat kebijakan yang dikelola pelanggan yang memberikan izin untuk membuat peran dengan izin tertentu.

   a. Pada panel navigasi, pilih Kebijakan, lalu pilih Buat kebijakan.

   b. Pada tab JSON ini, ganti konten placeholder dengan kebijakan berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ManageRolePoliciesForIDTGreengrass",
            "Effect": "Allow",
```

```
            "Action": [
                "iam:DetachRolePolicy",
                "iam:AttachRolePolicy"
            ],
            "Resource": [
                "arn:aws:iam::*:role/idt-*",
                "arn:aws:iam::*:role/GreengrassServiceRole"
            ],
            "Condition": {
                "ArnEquals": {
                    "iam:PolicyARN": [
                        "arn:aws:iam::aws:policy/service-role/
AWSGreengrassResourceAccessRolePolicy",
                        "arn:aws:iam::aws:policy/service-role/
GreengrassOTAUpdateArtifactAccess",
                        "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"
                    ]
                }
            }
        },
        {
            "Sid": "ManageRolesForIDTGreengrass",
            "Effect": "Allow",
            "Action": [
                "iam:CreateRole",
                "iam:DeleteRole",
                "iam:PassRole",
                "iam:GetRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/idt-*",
                "arn:aws:iam::*:role/GreengrassServiceRole"
            ]
        }
    ]
}
```

Important

Kebijakan berikut memberikan izin untuk membuat dan mengelola peran yang diperlukan oleh IDT untuk AWS IoT Greengrass. Ini termasuk izin untuk melampirkan berikut AWS kebijakan terkelola:

- AWSGreengrassResourceAccessRolePolicy
- GreenGrassota UpdateArtifactAccess
- AWSLambdaBasicExecutionRole

c. Pilih Berikutnya: Tanda.

d. Pilih Berikutnya: Tinjauan.

e. Untuk Nama, masukkan **`IDTGreengrassIAMPermissions`**. Di bawah Ringkasan, tinjau izin yang diberikan oleh kebijakan Anda.

f. Pilih Buat kebijakan.

3. Buat pengguna IAM dan lampirkan izin yang diperlukan oleh IDT untuk AWS IoT Greengrass.

a. Buat pengguna IAM. Ikuti langkah 1 hingga 5 di Membuat pengguna IAM (konsol) di Panduan Pengguna IAM.

b. Lampirkan izin untuk pengguna IAM Anda:

i. Pada halaman Setel izin, pilih Lampirkan kebijakan yang ada secara langsung.

ii. Cari kebijakan IDTGreengrassIAMPermissions yang Anda buat pada langkah sebelumnya. Pilih kotak centang.

iii. Cari AWSIoTDeviceTesterForGreengrassFullAccesskebijakan. Pilih kotak centang.

> **Note**
>
> AWSIoTDeviceTesterForGreengrassFullAccessIni adalah kebijakan AWS terkelola yang mendefinisikan izin yang diperlukan IDT untuk membuat dan mengakses AWS sumber daya yang digunakan untuk pengujian. Untuk informasi selengkapnya, lihat the section called “AWS kebijakan terkelola untuk IDT”.

c. Pilih Selanjutnya: Menandai.

d. Pilih Berikutnya: Tinjauan untuk melihat ringkasan pilihan Anda.

e. Pilih Buat pengguna.

f. Untuk melihat access key pengguna (access key ID dan secret access key), pilih Tampilkan di samping setiap kata sandi dan kunci akses rahasia. Untuk menyimpan kunci akses, pilih Download.csv lalu simpan file ke lokasi yang aman. Anda menggunakan informasi ini nanti untuk file kredensial AWS.

4. Langkah berikutnya: Konfigurasikan perangkat fisik.

Untuk mengonfigurasi izin untuk IDT (AWS CLI)

Ikuti langkah-langkah ini untuk menggunakan AWS CLI agar mengonfigurasi untuk IDT pada AWS IoT Greengrass. Jika Anda sudah mengonfigurasi izin di konsol, lewati ke the section called "Konfigurasikan perangkat Anda untuk menjalankan tes IDT" atau the section called "Opsional: Mengonfigurasi kontainer Docker".

1. Pada komputer Anda, instal dan konfigurasikan AWS CLI jika ia belum dipasang. Ikuti langkah-langkah di Menginstal AWS CLI di Panduan Pengguna AWS Command Line Interface.

   > **Note**
   >
   > AWS CLI adalah alat sumber terbuka yang dapat Anda gunakan untuk berinteraksi dengan layanan AWS dari shell baris-perintah Anda.

2. Buat kebijakan yang dikelola pelanggan yang memberikan izin untuk mengelola IDT dan AWS IoT Greengrass peran.

   Linux, macOS, or Unix

```
aws iam create-policy --policy-name IDTGreengrassIAMPermissions --policy-
document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ManageRolePoliciesForIDTGreengrass",
            "Effect": "Allow",
            "Action": [
                "iam:DetachRolePolicy",
                "iam:AttachRolePolicy"
            ],
            "Resource": [
                "arn:aws:iam::*:role/idt-*",
                "arn:aws:iam::*:role/GreengrassServiceRole"
            ],
            "Condition": {
                "ArnEquals": {
```

```
                    "iam:PolicyARN": [
                        "arn:aws:iam::aws:policy/service-role/
AWSGreengrassResourceAccessRolePolicy",
                        "arn:aws:iam::aws:policy/service-role/
GreengrassOTAUpdateArtifactAccess",
                        "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"
                    ]
                }
            }
        },
        {
            "Sid": "ManageRolesForIDTGreengrass",
            "Effect": "Allow",
            "Action": [
                "iam:CreateRole",
                "iam:DeleteRole",
                "iam:PassRole",
                "iam:GetRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/idt-*",
                "arn:aws:iam::*:role/GreengrassServiceRole"
            ]
        }
    ]
}'
```

Windows command prompt

```
aws iam create-policy --policy-name IDTGreengrassIAMPermissions --
policy-document '{\"Version\": \"2012-10-17\", \"Statement\": [{\"Sid
\": \"ManageRolePoliciesForIDTGreengrass\",\"Effect\": \"Allow\",
\"Action\": [\"iam:DetachRolePolicy\", \"iam:AttachRolePolicy\"],
 \"Resource\": [\"arn:aws:iam::*:role/idt-*\",\"arn:aws:iam::*:role/
GreengrassServiceRole\"],\"Condition\": {\"ArnEquals\": {\"iam:PolicyARN\":
 [\"arn:aws:iam::aws:policy/service-role/AWSGreengrassResourceAccessRolePolicy
\",\"arn:aws:iam::aws:policy/service-role/GreengrassOTAUpdateArtifactAccess
\",\"arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole\"]}}},
{\"Sid\": \"ManageRolesForIDTGreengrass\",\"Effect\": \"Allow\",\"Action\":
 [\"iam:CreateRole\",\"iam:DeleteRole\", \"iam:PassRole\", \"iam:GetRole
\"],\"Resource\": [\"arn:aws:iam::*:role/idt-*\",\"arn:aws:iam::*:role/
GreengrassServiceRole\"]}]}'
```

> **Note**
>
> Langkah ini mencakup contoh prompt perintah Windows karena menggunakan sintaks JSON yang berbeda dari perintah terminal Linux, MacOS, atau Unix.

3. Buat pengguna IAM dan lampirkan izin yang diperlukan oleh IDT untuk AWS IoT Greengrass.

   a. Buat pengguna IAM. Dalam contoh pengaturan ini, pengguna diberi nama `IDTGreengrassUser`.

   ```
   aws iam create-user --user-name IDTGreengrassUser
   ```

   b. Lampirkan kebijakan `IDTGreengrassIAMPermissions` yang Anda buat pada langkah 2 untuk pengguna IAM Anda. Ganti *<account-id>* dalam perintah dengan ID Akun AWS.

   ```
   aws iam attach-user-policy --user-name IDTGreengrassUser --policy-arn
    arn:aws:iam::<account-id>:policy/IDTGreengrassIAMPermissions
   ```

   c. Lampirkan `AWSIoTDeviceTesterForGreengrassFullAccess` kebijakan ini ke pengguna IAM.

   ```
   aws iam attach-user-policy --user-name IDTGreengrassUser --policy-arn
    arn:aws:iam::aws:policy/AWSIoTDeviceTesterForGreengrassFullAccess
   ```

   > **Note**
   >
   > AWSIoTDeviceTesterForGreengrassFullAccessIni adalah kebijakan AWS terkelola yang mendefinisikan izin yang diperlukan IDT untuk membuat dan mengakses AWS sumber daya yang digunakan untuk pengujian. Untuk informasi selengkapnya, lihat the section called “AWS kebijakan terkelola untuk IDT”.

4. Buat secret access key bagi pengguna.

   ```
   aws iam create-access-key --user-name IDTGreengrassUser
   ```

   Simpan output tersebut di lokasi yang aman. Anda menggunakan informasi ini nanti untuk mengonfigurasi file kredensial AWS.

5. Langkah selanjutnya: Konfigurasikan perangkat fisik.

AWS kebijakan terkelola untuk AWS IoT Tester Perangkat

Kebijakan AWSIoTDeviceTesterForGreengrassFullAccessterkelola memungkinkan IDT menjalankan operasi dan mengumpulkan metrik penggunaan. Kebijakan ini memberikan izin IDT berikut:

- `iot-device-tester:CheckVersion`. Periksa apakah satu set AWS IoT Greengrass, tes suite, dan IDT versi kompatibel.
- `iot-device-tester:DownloadTestSuite`. Unduh tes suite.
- `iot-device-tester:LatestIdt`. Dapatkan informasi tentang versi IDT terbaru yang tersedia untuk diunduh.
- `iot-device-tester:SendMetrics`. Publikasikan data penggunaan yang IDT kumpulkan tentang tes Anda.
- `iot-device-tester:SupportedVersion`. Terbitkan daftar AWS IoT Greengrass dan versi rangkaian pengujian yang didukung oleh IDT. Informasi ini ditampilkan di jendela baris perintah.

Konfigurasikan perangkat Anda untuk menjalankan tes IDT

Untuk mengonfigurasi perangkat Anda, Anda harus menginstal AWS IoT Greengrass dependensi, mengonfigurasi AWS IoT Greengrass Perangkat lunak Core, konfigurasikan komputer host Anda untuk mengakses perangkat Anda, dan konfigurasikan izin pengguna di perangkat Anda.

Verifikasi AWS IoT Greengrass depedensi pada perangkat yang sedang dites

Sebelum IDT untuk AWS IoT Greengrass dapat mengetes perangkat Anda, pastikan Anda telah menyiapkan perangkat seperti yang dijelaskan di Memulai dengan AWS IoT Greengrass. Untuk informasi tentang platform yang didukung, lihat Platform yang didukung.

Mengonfigurasi AWS IoT Greengrass perangkat lunak

IDT untuk AWS IoT Greengrass tes kompatibilitas perangkat Anda dengan versi AWS IoT Greengrass. IDT menyediakan dua pilihan untuk pengetesan AWS IoT Greengrass di perangkat Anda:

- Unduh dan gunakan versi AWS IoT Greengrass perangkat lunak Core. IDT menginstal perangkat lunak untuk Anda.
- Gunakan versi AWS IoT Greengrass erangkat lunak core telah diinstal pada perangkat Anda.

> **Note**
>
> Setiap versi AWS IoT Greengrass memiliki versi IDT yang sesuai. Anda harus mengunduh versi IDT yang sesuai dengan versi AWS IoT Greengrass yang Anda gunakan.

Bagian berikut menjelaskan opsi ini. Anda hanya butuh mengerjakan satu.

Opsi 1: MengunduhAWS IoT GreengrassPerangkat lunak intiAWS IoTDevice Tester untuk menggunakannya

Anda dapat mengunduh AWS IoT Greengrass perangkat lunak core dari AWS IoT Greengrass Perangkat lunak Core halaman unduhan.

1. Temukan arsitektur dan distribusi Linux yang benar, dan kemudian memilih Mengunduh.
2. Salin file tar.gz ke *<device-tester-extract-location>*`/products/greengrass/ggc`.

> **Note**
>
> Jangan mengubah nama file AWS IoT Greengrass tar.gz. Jangan menempatkan beberapa file dalam direktori ini untuk sistem operasi dan arsitektur yang sama. Sebagai contoh memiliki kedua `greengrass-linux-armv7l-1.7.1.tar.gz` dan `greengrass-linux-armv7l-1.8.1.tar.gz` file dalam direktori tersebut akan menyebabkan tes gagal.

Opsi 2: Gunakan instalasiAWS IoT GreengrassbersamaAWS IoTPenguji Perangkat

Mengonfigurasi IDT untuk mengetes AWS IoT Greengrass erangkat lunak Core yang diinstal di perangkat Anda dengan menambahkan `greengrassLocation` atribut untuk file `device.json` di folder *<device-tester-extract-location>*`/configs` ini. Misalnya:

```
"greengrassLocation" : "<path-to-greengrass-on-device>"
```

Untuk informasi lebih lanjut tentang file `device.json` ini, lihat Konfigurasikan device.json.

Pada perangkat Linux, lokasi default AWS IoT Greengrass perangkat lunak Core adalah `/greengrass`.

> **Note**
>
> Perangkat Anda harus memiliki instalasi AWS IoT Greengrass perangkat lunak Core yang belum dimulai.
> Pastikan Anda telah menambahkan `ggc_user` pengguna dan `ggc_group` pada perangkat Anda. Untuk informasi lebih lanjut, lihat pengaturan Lingkungan untuk AWS IoT Greengrass.

Konfigurasikan komputer host Anda untuk mengakses perangkat Anda yang sedang dites

IDT berjalan pada komputer host Anda dan harus dapat menggunakan SSH untuk terhubung ke perangkat Anda. Terdapat dua pilihan untuk memungkinkan IDT untuk mendapatkan akses SSH ke perangkat Anda yang diuji:

1. Ikuti petunjuk di sini untuk membuat pasangan kunci SSH dan otorisasi kunci Anda untuk masuk ke perangkat Anda yang sedang diuji tanpa menyebutkan kata sandi.
2. Berikan nama pengguna dan kata sandi untuk setiap perangkat di file `device.json`. Untuk informasi selengkapnya, lihat Konfigurasikan device.json.

Anda dapat menggunakan implementasi SSL apa pun untuk membuat kunci SSH. Petunjuk berikut menunjukkan cara menggunakan SSH-KEYGEN atau PuTTYgen (untuk Windows). Jika Anda menggunakan implementasi SSL lain, lihat dokumentasi untuk implementasi tersebut.

IDT menggunakan kunci SSH untuk diautentikasi dengan perangkat Anda yang sedang diuji.

Untuk membuat kunci SSH dengan SSH-KEYGEN

1. Buat kunci SSH

   Anda dapat menggunakan perintah ssh-keygen Open SSH untuk membuat pasangan kunci SSH. Jika Anda sudah memiliki pasanagn kunci SSH pada komputer host Anda, adalah praktik terbaik untuk membuat pasangan kunci SSH khusus untuk IDT. Dengan cara ini, setelah Anda menyelesaikan tes, komputer host Anda tidak dapat lagi terhubung ke perangkat Anda tanpa memasukkan kata sandi. Hal ini juga memungkinkan Anda membatasi akses ke perangkat jarak jauh hanya untuk yang membutuhkannya.

> **Note**
>
> Windows tidak memiliki klien SSH yang diinstal. Untuk informasi tentang cara menginstal klien SSH di Windows, lihat Unduh Perangkat Lunak Klien SSH.

Perintah ssh-keygen meminta Anda untuk memberikan nama dan path untuk menyimpan pasangan kunci tersebut. Secara default, file pasangan kunci diberi nama `id_rsa` (kunci privat) dan `id_rsa.pub` (kunci publik). Di macOS dan Linux, lokasi default file ini adalah `~/.ssh/`. Di Windows, lokasi default untuk file ini adalah `C:\Users\<user-name>\.ssh`.

Saat diminta, masukkan frase kunci untuk melindungi kunci SSH Anda. Untuk informasi lebih lanjut, lihat Buat Kunci SSH Baru.

2. Tambahkan kunci SSH yang diotorisasi pada perangkat Anda yang sedang diuji.

   IDT harus menggunakan kunci privat SSH Anda untuk masuk ke perangkat Anda yang sedang diuji. Untuk mengotorisasi kunci privat SSH Anda untuk masuk ke perangkat Anda yang sedang diuji, gunakan perintah ssh-copy-id dari komputer host Anda. Perintah ini menambahkan kunci publik Anda ke dalam file `~/.ssh/authorized_keys` pada perangkat Anda yang sedang diuji. Misalnya:

   $ ssh-copy-id ***<remote-ssh-user>***@***<remote-device-ip>***

   Di mana *`remote-ssh-user`* adalah nama pengguna yang digunakan untuk masuk ke perangkat Anda yang sedang diuji dan *`remote-device-ip`* adalah alamat IP perangkat yang sedang diuji untuk menjalankan tes terhadapnya. Misalnya:

   ssh-copy-id pi@192.168.1.5

   Saat diminta, masukkan kata sandi untuk nama pengguna yang Anda tentukan di perintah ssh-copy-id.

   ssh-copy-id mengasumsikan kunci publik tersebut bernama `id_rsa.pub` dan disimpan di lokasi default (pada macOS dan Linux, `~/.ssh/` dan pada Windows, `C:\Users\<user-name>\.ssh`). Jika Anda memberikan kunci publik nama yang berbeda atau menyimpannya di lokasi yang berbeda, Anda harus menentukan path yang memenuhi syarat untuk kunci publik SSH Anda dengan menggunakan -i untuk ssh-copy-id (misalnya, ssh-copy-id -i ~/my/path/

myKey.pub). Untuk informasi lebih lanjut tentang cara membuat kunci SSH dan menyalin kunci publik, lihat SSH-COPY-ID.

Untuk membuat kunci SSH dengan menggunakan PuTTYgen (hanya Windows)

1. Pastikan Anda mempunyai server dan klien OpenSSH yang terinstal pada perangkat Anda yang sedang diuji. Untuk informasi selengkapnya, lihat OpenSSH.
2. Instal PuTTYgen di perangkat Anda yang sedang diuji.
3. Buka PuTTYgen.
4. Pilih Buat dan gerakkan kursor mouse Anda di dalam kotak untuk menghasilkan kunci privat.
5. Dari menu Konversi, pilih Ekspor kunci OpenSSH, dan simpan kunci privat dengan ekstensi file `.pem`.
6. Tambahkan kunci publik ke file `/home/<user>/.ssh/authorized_keys` pada perangkat yang sedang diuji.
   a. Salin teks kunci publik dari jendela PuTTYgen.
   b. Gunakan PuTTY untuk membuat sesi pada perangkat Anda yang sedang diuji.
      i. Dari command prompt atau jendela Windows Powershell, jalankan perintah berikut:

         C:/***<path-to-putty>***/putty.exe -ssh ***<user>***@***<dut-ip-address>***

      ii. Saat diminta, masukkan kata sandi perangkat Anda.
      iii. Gunakan vi atau editor teks lain untuk menambahkan kunci publik ke file `/home/<user>/.ssh/authorized_keys` pada perangkat Anda yang sedang diuji.
7. Perbarui file `device.json` Anda dengan nama pengguna, alamat IP, dan path Anda ke file kunci privat yang baru saja Anda simpan di komputer host untuk setiap perangkat yang sedang diuji. Untuk informasi selengkapnya, lihat the section called “Konfigurasikan device.json”. Pastikan Anda memberikan path dan nama file yang lengkap untuk kunci privat dan gunakan garis miring ('/'). Misalnya, untuk path Windows `C:\DT\privatekey.pem`, gunakan `C:/DT/privatekey.pem` di file `device.json`.

Konfigurasikan izin pengguna di perangkat Anda

IDT melakukan operasi pada berbagai direktori dan file dalam perangkat yang diuji. Beberapa dari operasi ini memerlukan izin yang ditinggikan (menggunakan sudo). Untuk mengotomatisasi operasi

ini, IDT untuk AWS IoT Greengrass harus dapat menjalankan perintah dengan sudo tanpa diminta untuk kata sandi.

Ikuti langkah-langkah ini pada perangkat yang sedang dites untuk mengizinkan akses sudo tanpa diminta memasukkan kata sandi.

> **Note**
>
> `username` mengacu pada pengguna SSH yang digunakan oleh IDT untuk mengakses perangkat yang diuji.

Tambahkan pengguna ke grup sudo.

1. Pada perangkat yang sedang diuji, jalankan `sudo usermod -aG sudo` *`<username>`*.
2. Keluar, lalu masuk kembali agar perubahan diterapkan.
3. Untuk memverifikasi nama pengguna Anda telah berhasil ditambahkan, jalankan sudo echo test. Jika Anda tidak diminta untuk memasukkan kata sandi, pengguna Anda telah dikonfigurasi dengan benar.
4. Buka file `/etc/sudoers` dan tambahkan baris berikut ke akhir file:

   `<ssh-username>` `ALL=(ALL) NOPASSWD: ALL`

Konfigurasikan perangkat Anda untuk mengetes fitur opsional

Topik berikut menjelaskan cara mengonfigurasi perangkat Anda untuk menjalankan tes IDT untuk fitur opsional. Ikuti langkah-langkah konfigurasi ini hanya jika Anda ingin mengetes fitur ini. Jika tidak, lanjutkan ke the section called “Konfigurasikan pengaturan IDT”.

Topik

- Opsional: Mengonfigurasi kontainer Docker Anda untuk IDT untukAWS IoT Greengrass
- Opsional: Mengonfigurasi perangkat Anda untuk kualifikasi ML-nya

Opsional: Mengonfigurasi kontainer Docker Anda untuk IDT untukAWS IoT Greengrass

AWS IoT Greengrass menyediakan gambar Docker dan Dockerfile yang membuatnya lebih mudah untuk menjalankan AWS IoT Greengrass software Core dalam kontainer Docker. Setelah Anda mengatur AWS IoT Greengrass kontainer, Anda dapat menjalankan tes IDT. Saat ini, hanya x86_64 Docker arsitektur yang didukung untuk menjalankan IDT untuk AWS IoT Greengrass.

Fitur ini memerlukan IDT v2.3.0 atau yang lebih baru.

Proses pengaturan kontainer Docker untuk menjalankan tes IDT tergantung pada apakah Anda menggunakan gambar Docker atau Dockerfile yang disediakan oleh AWS IoT Greengrass.

- Gunakan gambar Docker. Gambar Docker memiliki AWS IoT Greengrass perangkat lunak Core dan dependensi terinstal.
- Menggunakan Dockerfile. Dockerfile berisi kode sumber yang dapat Anda gunakan untuk membangun citra kontainer AWS IoT Greengrass khusus. Citra dapat dimodifikasi untuk berjalan pada arsitektur platform yang berbeda atau untuk mengurangi ukuran citra.

> **Note**
>
> AWS IoT Greengrass tidak menyediakan gambar Dockerfiles atau Docker untuk AWS IoT Greengrass versi perangkat lunak inti 1.11.1. Untuk menjalankan tes IDT pada citra kontainer khusus milik Anda, citra Anda harus menyertakan dependensi didefinisikan dalam Dockerfile yang disediakan oleh AWS IoT Greengrass.

Fitur berikut tidak tersedia ketika Anda menjalankan AWS IoT Greengrass dalam kontainer Docker:

- Konektor yang berjalan di mode kontainer Greengrass ini. Untuk menjalankan konektor dalam kontainer Docker, konektor harus berjalan dengan mode Tanpa kontainer ini. Untuk menemukan konektor yang mendukung mode Tanpa kontainer ini, lihat the section called “AWS-disediakan konektor Greengrass”. Beberapa konektor ini memiliki parameter mode isolasi yang harus Anda atur ke Tanpa kontainer.
- Sumber daya perangkat dan volume lokal. Fungsi Lambda yang ditetapkan pengguna Anda yang berjalan dalam k Docker harus mengakses perangkat dan volume pada core secara langsung.

Mengonfigurasi gambar Docker yang disediakan oleh AWS IoT Greengrass

Ikuti langkah-langkah ini untuk mengonfigurasi AWS IoT Greengrass gambar Docker untuk menjalankan tes IDT.

Prasyarat

Sebelum Anda mulai tutorial ini, Anda harus melakukan hal berikut.

- Anda harus menginstal perangkat lunak dan versi berikut pada komputer host Anda berdasarkan versi AWS Command Line Interface (AWS CLI) yang Anda pilih.

  AWS CLI version 2

  - Docker versi 18.09 atau yang lebih baru. Versi sebelumnya juga dapat berfungsi, namun kami merekomendasikan 18.09 atau yang lebih baru.
  - AWS CLI versi 2.0.0 atau yang lebih baru.
    - Untuk menginstal AWS CLI versi 2, lihat Menginstal AWS CLI versi 2.
    - Untuk mengonfigurasi AWS CLI, lihat Mengonfigurasi AWS CLI.

    > **Note**
    >
    > Untuk meningkatkan AWS CLI versi 2 ke yang lebih baru pada komputer Windows, Anda harus mengulangi proses instalasi MSI ini.

  AWS CLI version 1

  - Docker versi 18.09 atau yang lebih baru. Versi sebelumnya juga dapat berfungsi, namun kami merekomendasikan 18.09 atau yang lebih baru.
  - Python versi 3.6 atau lebih baru.
  - pip versi 18.1 atau yang lebih baru.
  - AWS CLI versi 1.17.10 atau yang lebih baru
    - Untuk menginstal AWS CLI versi 1, lihat Menginstal AWS CLI versi 1.
    - Untuk mengonfigurasi AWS CLI, lihat Mengonfigurasi AWS CLI.
    - Untuk meningkatkan AWS CLI versi 1 ke versi terbaru, jalankan perintah berikut.

      ```
      pip install awscli --upgrade --user
      ```

> **Note**
>
> Jika Anda menggunakan Instalasi MSI dari AWS CLI versi 1 pada Windows, perhatikan hal-hal berikut:
>
> - Jika instalasi AWS CLI versi 1 gagal untuk menginstal botocore, coba gunakan Instalasi Python dan pip.
> - Untuk meningkatkan AWS CLI versi 1 ke yang lebih baru, anda mesti mengulangi proses instalasi MSI.

- Untuk mengakses sumber daya Amazon Elastic Container Registry (Amazon ECR), Anda harus memberikan izin berikut.
  - Amazon ECR mengharuskan pengguna memiliki izin `ecr:GetAuthorizationToken` melalui kebijakan (IAM) AWS Identity and Access Management sebelum mereka dapat mengautentikasi ke registrasi dan mendorong atau menarik citra dari repositori Amazon ECR. Untuk informasi lebih lanjut, lihat Contoh Kebijakan Repositori Amazon ECR dan Mengakses Satu Repositori Amazon ECR di Amazon Elastic Container Registry.

1. Unduh gambar Docker dan konfiguras kontainer. Anda dapat mengunduh citra prebuilt dari Docker Hub atau Amazon Elastic Container Registry (Amazon ECR) dan menjalankannya pada platform Windows, MacOS, dan Linux (x86_64).

   Untuk mengunduh gambar Docker dari Amazon ECR, selesaikan semua langkah di the section called “Dapatkan citra kontainer AWS IoT Greengrass dari Amazon ECR”. Kemudian, kembali ke topik ini untuk melanjutkan konfigurasi.
2. Hanya pengguna Linux: Pastikan pengguna yang menjalankan IDT memiliki izin untuk menjalankan perintah Docker. Untuk informasi lebih lanjut, lihat Mengelola Docker sebagai pengguna non-root dalam dokumentasi Docker.
3. Untuk menjalankan AWS IoT Greengrass kontainer, gunakan perintah untuk sistem operasi Anda:

   Linux

```
docker run --rm --init -it -d --name aws-iot-greengrass \
-p 8883:8883 \
```

```
-v <host-path-to-kernel-config-file>:<container-path> \
<image-repository>:<tag>
```

- Ganti *<host-path-to-kernel-config-file>* dengan jalur menuju file konfigurasi kernel pada host dan *<container-path>* dengan jalur di mana volume terpasang dalam kontainer.

  File konfigurasi kernel pada host biasanya terletak di `/proc/config.gz` atau `/boot/config-`*<kernel-release-date>*. Anda dapat menjalankan `uname -r` untuk menemukan nilai *<kernel-release-date>* ini.

  Contoh: Untuk memasang file konfigurasi dari`/boot/config-`*<kernel-release-date>*

  ```
  -v /boot/config-4.15.0-74-generic:/boot/config-4.15.0-74-generic \
  ```

  Contoh: Untuk memasang file konfigurasi dari`proc/config.gz`

  ```
  -v /proc/config.gz:/proc/config.gz \
  ```

- Ganti *<image-repository>*:*<tag>* dalam perintah dengan nama repositori dan menandai dari citra target.

  Contoh: Untuk menunjuk ke versi terbaruAWS IoT GreengrassPerangkat lunak inti

  ```
  216483018798.dkr.ecr.us-west-2.amazonaws.com/aws-iot-greengrass:latest
  ```

  Untuk mendapatkan daftar AWS IoT Greengrass gambar Docker, jalankan perintah berikut.

  ```
  aws ecr list-images --region us-west-2 --registry-id 216483018798 --
  repository-name aws-iot-greengrass
  ```

macOS

```
docker run --rm --init -it -d --name aws-iot-greengrass \
-p 8883:8883 \
<image-repository>:<tag>
```

- Ganti *<image-repository>*:*<tag>* dalam perintah dengan nama repositori dan menandai dari citra target.

  Contoh: Untuk menunjuk ke versi terbaruAWS IoT GreengrassPerangkat lunak inti

  ```
  216483018798.dkr.ecr.us-west-2.amazonaws.com/aws-iot-greengrass:latest
  ```

  Untuk mendapatkan daftar AWS IoT Greengrass gambar Docker, jalankan perintah berikut:

  ```
  aws ecr list-images --region us-west-2 --registry-id 216483018798 --
  repository-name aws-iot-greengrass
  ```

Windows

```
docker run --rm --init -it -d --name aws-iot-greengrass \
-p 8883:8883 \
<image-repository>:<tag>
```

- Ganti *<image-repository>*:*<tag>* dalam perintah dengan nama repositori dan menandai dari citra target.

  Contoh: Untuk menunjuk ke versi terbaruAWS IoT GreengrassPerangkat lunak inti

  ```
  216483018798.dkr.ecr.us-west-2.amazonaws.com/aws-iot-greengrass:latest
  ```

  Untuk mendapatkan daftar AWS IoT Greengrass gambar Docker, jalankan perintah berikut:

  ```
  aws ecr list-images --region us-west-2 --registry-id 216483018798 --
  repository-name aws-iot-greengrass
  ```

Important

Ketika pengujian dengan IDT, jangan sertakan argumen `--entrypoint /greengrass-entrypoint.sh \` yang digunakan untuk menjalankan citra untuk penggunaan AWS IoT Greengrass umum.

4. Langkah berikutnya: MengkonfigurasiAWSkredenensi dan`device.jsonfail`.

Konfigurasikan dockerfile yang disediakan oleh AWS IoT Greengrass

Ikuti langkah-langkah ini untuk mengonfigurasi gambar Docker yang dibangun dari AWS IoT Greengrass Dockerfile untuk menjalankan tes IDT.

1. Dari the section called “AWS IoT Greengrass Perangkat lunak Docker”, mengunduh paket Dockerfile ke komputer host Anda dan mengekstraksi itu.
2. Buka `README.md`. Tiga langkah selanjutnya mengacu pada bagian dalam file ini.
3. Pastikan Anda memenuhi persyaratan di bagian Prasyarat.
4. Hanya pengguna Linux: LengkapiAktifkan Symlink dan Perlindungan HardlinkdanMengaktifkan Penerusan Jaringan IPv4langkah-langkah.
5. Untuk membuat gambar Docker, selesaikan semua langkah di Langkah 1. Bangun AWS IoT Greengrass Gambar Docker. Kemudian, kembali ke topik ini untuk melanjutkan konfigurasi.
6. Untuk menjalankan AWS IoT Greengrass kontainer, gunakan perintah untuk sistem operasi Anda:

   Linux

   ```
   docker run --rm --init -it -d --name aws-iot-greengrass \
   -p 8883:8883 \
   -v <host-path-to-kernel-config-file>:<container-path> \
   <image-repository>:<tag>
   ```

   - Ganti *<host-path-to-kernel-config-file>* dengan jalur menuju file konfigurasi kernel pada host dan *<container-path>* dengan jalur di mana volume terpasang dalam kontainer.

     File konfigurasi kernel pada host biasanya terletak di `/proc/config.gz` atau `/boot/config-`*<kernel-release-date>*. Anda dapat menjalankan `uname -r` untuk menemukan nilai *<kernel-release-date>* ini.

     Contoh: Untuk memasang file konfigurasi dari`/boot/config-`*<kernel-release-date>*

     ```
     -v /boot/config-4.15.0-74-generic:/boot/config-4.15.0-74-generic \
     ```

Contoh: Untuk memasang file konfigurasi dari`proc/config.gz`

```
-v /proc/config.gz:/proc/config.gz \
```

- Ganti *<image-repository>*:*<tag>* dalam perintah dengan nama repositori dan menandai dari citra target.

  Contoh: Untuk menunjuk ke versi terbaruAWS IoT GreengrassPerangkat lunak inti

  ```
  216483018798.dkr.ecr.us-west-2.amazonaws.com/aws-iot-greengrass:latest
  ```

  Untuk mendapatkan daftar AWS IoT Greengrass gambar Docker, jalankan perintah berikut.

  ```
  aws ecr list-images --region us-west-2 --registry-id 216483018798 --
  repository-name aws-iot-greengrass
  ```

macOS

```
docker run --rm --init -it -d --name aws-iot-greengrass \
-p 8883:8883 \
<image-repository>:<tag>
```

- Ganti *<image-repository>*:*<tag>* dalam perintah dengan nama repositori dan menandai dari citra target.

  Contoh: Untuk menunjuk ke versi terbaruAWS IoT GreengrassPerangkat lunak inti

  ```
  216483018798.dkr.ecr.us-west-2.amazonaws.com/aws-iot-greengrass:latest
  ```

  Untuk mendapatkan daftar AWS IoT Greengrass gambar Docker, jalankan perintah berikut:

  ```
  aws ecr list-images --region us-west-2 --registry-id 216483018798 --
  repository-name aws-iot-greengrass
  ```

Windows

```
docker run --rm --init -it -d --name aws-iot-greengrass \
```

```
-p 8883:8883 \
<image-repository>:<tag>
```

- Ganti *<image-repository>*:*<tag>* dalam perintah dengan nama repositori dan menandai dari citra target.

  Contoh: Untuk menunjuk ke versi terbaruAWS IoT GreengrassPerangkat lunak inti

  ```
  216483018798.dkr.ecr.us-west-2.amazonaws.com/aws-iot-greengrass:latest
  ```

  Untuk mendapatkan daftar AWS IoT Greengrass gambar Docker, jalankan perintah berikut:

  ```
  aws ecr list-images --region us-west-2 --registry-id 216483018798 --
  repository-name aws-iot-greengrass
  ```

> **Important**
>
> Ketika pengujian dengan IDT, jangan sertakan argumen `--entrypoint /greengrass-entrypoint.sh \` yang digunakan untuk menjalankan citra untuk penggunaan AWS IoT Greengrass umum.

7. Langkah berikutnya: MengkonfigurasiAWSkredenensi dan`device.json`fail.

Memecahkan masalah pengaturan kontainer Docker untuk IDT untuk AWS IoT Greengrass

Gunakan informasi berikut untuk membantu memecahkan masalah dengan menjalankan kontainer Docker untuk IDT untuk AWS IoT Greengrass pengetesan.

PERINGATAN: Kesalahan ketika memuat config: /home/user/.docker/config.json - stat /home/<user>/.docker/config.json: izin ditolak

Jika Anda mendapatkan kesalahan ini ketika menjalankan perintah `docker` di Linux, jalankan perintah berikut. Ganti *<user>* dalam perintah berikut dengan pengguna yang menjalankan IDT.

```
sudo chown <user>:<user> /home/<user>/.docker -R
sudo chmod g+rwx /home/<user>/.docker -R
```

Opsional: Mengonfigurasi perangkat Anda untuk kualifikasi ML-nya

IDT untuk AWS IoT Greengrass menyediakan tes kualifikasi machine learning (ML) untuk memvalidasi bahwa perangkat Anda dapat melakukan inferensi ML secara lokal menggunakan model cloud-trained.

Untuk menjalankan tes kualifikasi ML, Anda harus terlebih dahulu mengonfigurasi perangkat Anda seperti yang dijelaskan di the section called “Konfigurasikan perangkat Anda untuk menjalankan tes IDT”. Kemudian, ikuti langkah-langkah dalam topik ini untuk menginstal dependensi untuk kerangka kerja ML yang Anda ingin jalankan.

IDT v3.1.0 atau yang lebih baru diperlukan untuk menjalankan tes untuk kualifikasi ML.

Menginstal dependensi kerangka kerja ML

Semua dependensi kerangka kerja ML harus diinstal di bawah direktori `/usr/local/lib/python3.x/site-packages` ini. Untuk memastikan mereka dipasang di direktori yang benar, kami merekomendasikan Anda menggunakan `sudo` izin root ketika menginstal dependensi. Lingkungan virtual tidak didukung untuk tes kualifikasi.

> **Note**
>
> Jika anda sedang mengetes fungsi Lambda yang berjalan dengan kontainerisasi (di mode kontainer Greengrass ini), membuat symlink untuk perpustakaan Python di bawah `/usr/local/lib/python3.x` tidak didukung. Untuk menghindari kesalahan, Anda harus menginstal dependensi di bawah direktori yang benar.

Ikuti langkah-langkah untuk menginstal dependensi untuk kerangka kerja target Anda:

- Menginstal dependensi MXNet
- the section called “Pasang TensorFlow dependensi”
- Instal dependensi DLR

Instal dependensi Apache MXNet

Tes kualifikasi IDT untuk kerangka kerja ini memiliki dependensi sebagai berikut:

- Python 3.6 atau Python 3.7.

> **Note**
>
> Jika Anda menggunakan Python 3.6, Anda harus membuat sebuah tautan simbolik dari binari Python 3.7 ke Python 3.6. Ini mengonfigurasi perangkat Anda untuk memenuhi persyaratan Python untuk AWS IoT Greengrass. Misalnya:
>
> ```
> sudo ln -s path-to-python-3.6/python3.6 path-to-python-3.7/python3.7
> ```

- Apache MXNet v1.2.1 atau yang lebih baru.
- NumPy. Versi harus kompatibel dengan versi MXNet Anda.

Menginstal MXNet

Ikuti instruksi di dokumentasi MXNet untuk instal MXNet.

> **Note**
>
> Jika Python 2.x dan Python 3.x keduanya diinstal pada perangkat Anda, gunakan Python 3.x dalam perintah yang Anda jalankan untuk menginstal dependensi.

Memvalidasi instalasi MXNet

Memilih salah satu opsi berikut untuk memvalidasi instalasi MXNet.

Opsi 1: SSH ke perangkat Anda dan jalankan skrip

1. SSH ke perangkat Anda.
2. Jalankan skrip berikut untuk memverifikasi bahwa dependensi diinstal dengan benar.

   ```
   sudo python3.7 -c "import mxnet; print(mxnet.__version__)"
   ```

   ```
   sudo python3.7 -c "import numpy; print(numpy.__version__)"
   ```

   Output mencetak nomor versi dan skrip harus keluar tanpa kesalahan.

Opsi 2: Menjalankan tes depedensi IDT

1. Pastikan bahwa `device.json` dikonfigurasi untuk kualifikasi ML. Untuk informasi selengkapnya, lihat the section called “Konfigurasikan device.json untuk kualifikasi ML”.
2. Jalankan tes dependensi untuk kerangka kerja.

   ```
   devicetester_[linux | mac | win_x86-64] run-suite --group-id mldependencies --test-id mxnet_dependency_check
   ```

   Ringkasan tes menampilkan `PASSED` hasil untuk `mldependencies`.

Pasang TensorFlow dependensi

Tes kualifikasi IDT untuk kerangka kerja ini memiliki dependensi sebagai berikut:

- Python 3.6 atau Python 3.7.

  > **Note**
  >
  > Jika Anda menggunakan Python 3.6, Anda harus membuat sebuah tautan simbolik dari binari Python 3.7 ke Python 3.6. Ini mengonfigurasi perangkat Anda untuk memenuhi persyaratan Python untuk AWS IoT Greengrass. Misalnya:
  >
  > ```
  > sudo ln -s path-to-python-3.6/python3.6 path-to-python-3.7/python3.7
  > ```

- TensorFlow 1.x.

Menginstal TensorFlow

Ikuti instruksi di TensorFlow dokumentasi untuk menginstal TensorFlow 1.xdengan pipatau dari sumber.

> **Note**
>
> Jika Python 2.x dan Python 3.x keduanya diinstal pada perangkat Anda, gunakan Python 3.x dalam perintah yang Anda jalankan untuk menginstal dependensi.

Memvalidasi TensorFlow instalasi

Pilih salah satu opsi berikut untuk memvalidasi TensorFlow instalasi.

Opsi 1: SSH ke perangkat Anda dan jalankan skrip

1. SSH ke perangkat Anda.
2. Menjalankan skrip berikut untuk memverifikasi bahwa depedensi telah diinstal dengan benar.

   ```
   sudo python3.7 -c "import tensorflow; print(tensorflow.__version__)"
   ```

   Output mencetak nomor versi dan skrip harus keluar tanpa kesalahan.

Opsi 2: Menjalankan tes depedensi IDT

1. Pastikan bahwa `device.json` dikonfigurasi untuk kualifikasi ML. Untuk informasi selengkapnya, lihat the section called “Konfigurasikan device.json untuk kualifikasi ML”.
2. Jalankan tes dependensi untuk kerangka kerja.

   ```
   devicetester_[linux | mac | win_x86-64] run-suite --group-id mldependencies --test-
   id tensorflow_dependency_check
   ```

   Ringkasan tes menampilkan `PASSED` hasil untuk `mldependencies`.

Instal Amazon SageMaker Deep Learning Runtime (DLR) dependensi

Tes kualifikasi IDT untuk kerangka kerja ini memiliki dependensi sebagai berikut:

- Python 3.6 atau Python 3.7.

> **Note**
>
> Jika Anda menggunakan Python 3.6, Anda harus membuat sebuah tautan simbolik dari binari Python 3.7 ke Python 3.6. Ini mengonfigurasi perangkat Anda untuk memenuhi persyaratan Python untuk AWS IoT Greengrass. Misalnya:

```
sudo ln -s path-to-python-3.6/python3.6 path-to-python-3.7/python3.7
```

- SageMaker Neo DLR.
- numpy.

Setelah Anda menginstal dependensi tes DLR, Anda harus mengkompilasi model.

Menginstal DLR

Ikuti instruksi di dokumentasi DLR untuk instal Neo DLR.

> **Note**
>
> Jika Python 2.x dan Python 3.x keduanya diinstal pada perangkat Anda, gunakan Python 3.x dalam perintah yang Anda jalankan untuk menginstal dependensi.

Memvalidasi instalasi DLR

Pilih salah satu opsi berikut untuk memvalidasi instalasi DLR.

Opsi 1: SSH ke perangkat Anda dan jalankan skrip

1. SSH ke perangkat Anda.
2. Jalankan skrip berikut untuk memverifikasi bahwa dependensi diinstal dengan benar.

   ```
   sudo python3.7 -c "import dlr; print(dlr.__version__)"
   ```

   ```
   sudo python3.7 -c "import numpy; print(numpy.__version__)"
   ```

   Output mencetak nomor versi dan skrip harus keluar tanpa kesalahan.

Opsi 2: Menjalankan tes depedensi IDT

1. Pastikan bahwa `device.json` dikonfigurasi untuk kualifikasi ML. Untuk informasi selengkapnya, lihat the section called “Konfigurasikan device.json untuk kualifikasi ML”.
2. Jalankan tes dependensi untuk kerangka kerja.

```
devicetester_[linux | mac | win_x86-64] run-suite --group-id mldependencies --test-
id dlr_dependency_check
```

Ringkasan tes menampilkan `PASSED` hasil untuk `mldependencies`.

Kompilasi model DLR

Anda harus mengkompilasi model DLR sebelum Anda dapat menggunakannya untuk tes kualifikasi ML. Pilih langkah-langkah, memilih salah satu opsi berikut.

Opsi 1: Gunakan Amazon SageMaker untuk mengkompilasi model

Ikuti langkah-langkah berikut untuk menggunakan SageMaker untuk mengkompilasi model ML-yang disediakan oleh IDT. Model ini sudah dilatih dengan Apache MXNet.

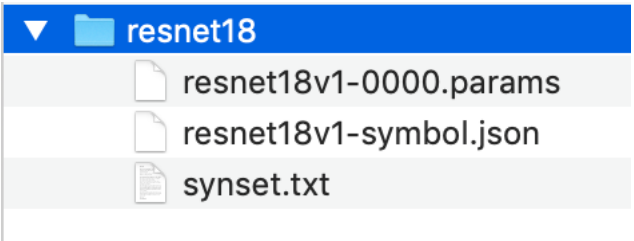
1. Verifikasi bahwa jenis perangkat Anda didukung oleh SageMaker. Untuk informasi selengkapnya, lihatopsi perangkatsangAmazon SageMaker Referensi API. Jika jenis perangkat Anda saat ini tidak didukung oleh SageMaker, ikuti langkah-langkah di the section called “Opsi 2: Gunakan TVM untuk mengompilasi model DLR”.

   > **Note**
   >
   > Menjalankan tes DLR dengan model yang disusun oleh SageMaker mungkin memakan waktu 4 atau 5 menit. Jangan hentikan IDT selama waktu ini.

2. Mengunduh file tarball yang berisi model MXNet yang belum dikompilasi dan telah terlatih untuk DLR:

   - dlr-noncompiled-model-1.0.tar.gz

3. Dekompresi tarball. Perintah ini menghasilkan struktur direktori berikut.

4. Pindahkan `synset.txt` keluar dari `resnet18` direktori. Buat catatan tentang lokasi baru. Anda salin file ini ke direktori model dikompilasi yang lebih baru..

5. Kompres konten `resnet18` direktori.

```
tar cvfz model.tar.gz resnet18v1-symbol.json resnet18v1-0000.params
```

6. Unggah file terkompresi ke bucket Amazon S3 di Akun AWS, dan kemudian ikuti langkah-langkah dalam Mengkompilasi Model (Konsol) untuk membuat pekerjaan kompilasi.

   a. Untuk Konfigurasi input, gunakan nilai berikut:

      - Untuk Konfigurasi input data, memasukkan `{"data": [1, 3, 224, 224]}`.
      - Untuk Kerangka kerja machine learning, memilih `MXNet`.

   b. Untuk Konfigurasi output, gunakan nilai berikut:

      - Untuk Lokasi output S3, memasukkan jalur ke bucket atau folder Amazon S3 di mana Anda ingin menyimpan model yang dikompilasi.
      - Untuk Perangkat target, pilih jenis perangkat Anda.

7. Mengunduh model yang dikompilasi dari lokasi output yang Anda tentukan, dan kemudian unzip file.
8. Salin `synset.txt` ke dalam direktori model yang dikompilasi.
9. Ubah nama direktori model yang dikompilasi menjadi `resnet18`.

   Direktori model terkompilasi Anda harus memiliki struktur direktori berikut.

Opsi 2: Gunakan TVM untuk mengompilasi model DLR

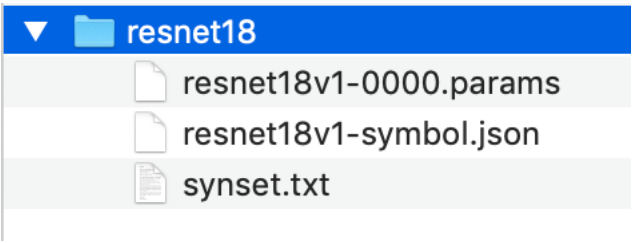
Ikuti langkah-langkah berikut untuk menggunakan TVM untuk mengompilasi model ML yang disediakan oleh IDT. Model ini sudah dilatih dengan Apache MXNet, sehingga Anda harus menginstal MXNet pada komputer atau perangkat di mana Anda mengkompilasi model. Untuk menginstal MXNet, ikuti petunjuk di dokumentasi MXNet.

> **Note**
>
> Kami merekomendasikan bahwa Anda mengkompilasi model pada perangkat target Anda. Praktek ini opsional, tetapi dapat membantu memastikan kompatibilitas dan mengurangi potensi masalah.

1. Mengunduh file tarball yang berisi model MXNet yang belum dikompilasi dan telah terlatih untuk DLR:

   - dlr-noncompiled-model-1.0.tar.gz

2. Dekompresi tarball. Perintah ini menghasilkan struktur direktori berikut.

3. Ikuti instruksi di dokumentasi TVM untuk membangun dan menginstal TVM dari sumber untuk platform Anda.

4. Setelah TVM dibangun, jalankan kompilasi TVM untuk model resnet18. Langkah-langkah berikut didasarkan pada Tutorial Quick Start untuk Mengkompilasi Model Deep Learning dalam dokumentasi TVM.

   a. Buka `relay_quick_start.py` file dari repositori TVM kloning.

   b. Update kode yang mendefinisikan jaringan syaraf tiruan dalam relay. Anda dapat menggunakan salah satu opsi berikut:

      - Opsi 1: Gunakan`mxnet.gluon.model_zoo.vision.get_model`untuk mendapatkan modul relay dan parameter:

        ```
        from mxnet.gluon.model_zoo.vision import get_model
        block = get_model('resnet18_v1', pretrained=True)
        mod, params = relay.frontend.from_mxnet(block, {"data": data_shape})
        ```

- Opsi 2: Dari model yang tidak dikompilasi yang Anda unduh pada langkah 1, salin file berikut ke direktori yang sama dengan`relay_quick_start.py`berkas. File-file ini berisi modul relay dan parameter.
  - `resnet18v1-symbol.json`
  - `resnet18v1-0000.params`

c. Update kode yang menyimpan dan memuat modul yang dikompilasi untuk menggunakan kode berikut.

```
from tvm.contrib import util
path_lib = "deploy_lib.so"
#  Export the model library based on your device architecture
lib.export_library("deploy_lib.so", cc="aarch64-linux-gnu-g++")
with open("deploy_graph.json", "w") as fo:
    fo.write(graph)
with open("deploy_param.params", "wb") as fo:
    fo.write(relay.save_param_dict(params))
```

d. Bangun model:

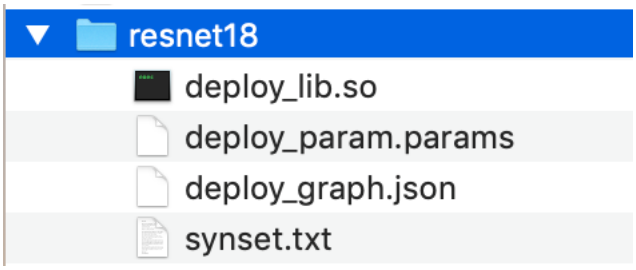
```
python3 tutorials/relay_quick_start.py --build-dir ./model
```

Perintah ini menghasilkan file-file berikut.

- `deploy_graph.json`
- `deploy_lib.so`
- `deploy_param.params`

5. Salin berkas model yang dihasilkan ke dalam direktori bernama `resnet18`. Ini adalah direktori model terkompilasi Anda.
6. Salin direktori model yang dikompilasi ke komputer host Anda. Kemudian salin `synset.txt` dari model yang tidak dikompilasi yang Anda unduh pada langkah 1 ke dalam direktori model yang dikompilasi.

   Direktori model terkompilasi Anda harus memiliki struktur direktori berikut.



Selanjutnya, konfigurasi kredensial AWS Anda dan `device.json` file.

Konfigurasikan pengaturan IDT untuk menjalankan rangkaian kualifikasi AWS IoT Greengrass

Sebelum menjalankan tes, Anda harus mengkonfigurasi pengaturan untuk AWS kredensial dan perangkat di komputer host Anda.

Konfigurasikan kredensial AWS Anda

Anda harus mengonfigurasi kredensial pengguna IAM Anda di file *<device-tester-extract-location>* `/configs/config.json` ini. Gunakan kredensial untuk IDT untuk AWS IoT Greengrass pengguna yang dibuat di the section called “Buat dan konfigurasi pengguna Akun AWS”. Anda dapat menentukan kredensial Anda dengan salah satu dari dua cara berikut:

- File kredensial
- Variabel lingkungan

Konfigurasikan AWS kredensial dengan file kredensial

IDT menggunakan file kredensials yang sama sebagai AWS CLI. Untuk informasi selengkapnya, lihat File konfigurasi dan kredensial.

Lokasi file kredensial itu bervariasi, tergantung pada sistem operasi yang Anda gunakan:

- macOS, Linux: `~/.aws/credentials`
- Windows: `C:\Users\`*`UserName`*`\.aws\credentials`

Tambahkan kredensial AWS Anda ke file `credentials` dalam format berikut:

```
[default]
```

```
aws_access_key_id = <your_access_key_id>
aws_secret_access_key = <your_secret_access_key>
```

Untuk mengonfigurasi IDT untuk AWS IoT Greengrass untuk menggunakan AWS kredensial dari file `credentials` Anda, edit file `config.json` Anda sebagai berikut:

```
{
 "awsRegion": "us-west-2",
 "auth": {
  "method": "file",
  "credentials": {
   "profile": "default"
  }
 }
}
```

> **Note**
>
> Jika Anda tidak menggunakan `default` AWS profile, pastikan untuk mengubah nama profil di file `config.json` Anda. Untuk informasi selengkapnya, lihat Profil Bernama.

Konfigurasi kredensial AWS dengan variabel lingkungan

Variabel lingkungan adalah variabel yang dikelola oleh sistem operasi dan digunakan oleh perintah sistem. Mereka tidak disimpan jika Anda menutup sesi SSH. IDT untuk AWS IoT Greengrass dapat menggunakan `AWS_ACCESS_KEY_ID` dan `AWS_SECRET_ACCESS_KEY` variabel lingkungan untuk menyimpan AWS kredensial.

Untuk mengatur variabel ini di Linux, macOS, atau Unix, gunakan export:

```
export AWS_ACCESS_KEY_ID=<your_access_key_id>
export AWS_SECRET_ACCESS_KEY=<your_secret_access_key>
```

Untuk menetapkan variabel ini di Windows, gunakan set:

```
set AWS_ACCESS_KEY_ID=<your_access_key_id>
set AWS_SECRET_ACCESS_KEY=<your_secret_access_key>
```

Untuk mengonfigurasi IDT untuk menggunakan variabel lingkungan, edit bagian `auth` di file `config.json` Anda. Berikut ini contohnya:

```
{
 "awsRegion": "us-west-2",
 "auth": {
  "method": "environment"
 }
}
```

Konfigurasikan device.json

Selain AWS kredensial, IDT untuk AWS IoT Greengrass membutuhkan informasi tentang perangkat yang tes dijalankan pada (misalnya, alamat IP, informasi login, sistem operasi, dan arsitektur CPU).

Anda harus memberikan informasi ini menggunakan templat `device.json` yang terletak di *<device_tester_extract_location>*`/configs/device.json`:

Physical device

```
[
  {
    "id": "<pool-id>",
    "sku": "<sku>",
    "features": [
      {
        "name": "os",
        "value": "linux | ubuntu | openwrt"
      },
      {
        "name": "arch",
        "value": "x86_64 | armv6l | armv7l | aarch64"
      },
      {
        "name": "container",
        "value": "yes | no"
      },
      {
        "name": "docker",
        "value": "yes | no"
      },
      {
        "name": "streamManagement",
```

```
        "value": "yes | no"
      },
      {
        "name": "hsi",
        "value": "yes | no"
      },
      {
        "name": "ml",
        "value": "mxnet | tensorflow | dlr | mxnet,dlr,tensorflow | no"
      },
      ********** Remove the section below if the device is not qualifying for ML
**************,
      {
        "name": "mlLambdaContainerizationMode",
        "value": "container | process | both"
      },
      {
        "name": "processor",
        "value": "cpu | gpu"
      },

*******************************************************************************************
   ],
   ********** Remove the section below if the device is not qualifying for HSI
***************
   "hsm": {
     "p11Provider": "/path/to/pkcs11ProviderLibrary",
     "slotLabel": "<slot_label>",
     "slotUserPin": "<slot_pin>",
     "privateKeyLabel": "<key_label>",
     "openSSLEngine": "/path/to/openssl/engine"
   },

*******************************************************************************************
   ********** Remove the section below if the device is not qualifying for ML
****************
   "machineLearning": {
     "dlrModelPath": "/path/to/compiled/dlr/model",
     "environmentVariables": [
       {
         "key": "<environment-variable-name>",
         "value": "<Path:$PATH>"
       }
     ],
```

```
      "deviceResources": [
        {
          "name": "<resource-name>",
          "path": "<resource-path>",
          "type": "device | volume"
        }
      ]
    },

 ********************************************************************************************
    "kernelConfigLocation": "",
    "greengrassLocation": "",
    "devices": [
      {
        "id": "<device-id>",
        "connectivity": {
          "protocol": "ssh",
          "ip": "<ip-address>",
          "port": 22,
          "auth": {
            "method": "pki | password",
            "credentials": {
              "user": "<user-name>",
              "privKeyPath": "/path/to/private/key",
              "password": "<password>"
            }
          }
        }
      }
    ]
  }
]
```

> **Note**
>
> Tentukan `privKeyPath` hanya jika `method` diatur ke `pki`.
> Tentukan `password` hanya jika `method` diatur ke `password`.

Docker container

```
[
```

```
 {
   "id": "<pool-id>",
   "sku": "<sku>",
   "features": [
     {
       "name": "os",
       "value": "linux | ubuntu | openwrt"
     },
     {
       "name": "arch",
       "value": "x86_64"
     },
     {
       "name": "container",
       "value": "no"
     },
     {
       "name": "docker",
       "value": "no"
     },
     {
       "name": "streamManagement",
       "value": "yes | no"
     },
     {
       "name": "hsi",
       "value": "no"
     },
     {
       "name": "ml",
       "value": "mxnet | tensorflow | dlr | mxnet,dlr,tensorflow | no"
     },
     ********** Remove the section below if the device is not qualifying for ML
**************,
     {
       "name": "mlLambdaContainerizationMode",
       "value": "process"
     },
     {
       "name": "processor",
       "value": "cpu | gpu"
     },

**********************************************************************************
```

```
    ],
    ********** Remove the section below if the device is not qualifying for ML
 ***************
    "machineLearning": {
      "dlrModelPath": "/path/to/compiled/dlr/model",
      "environmentVariables": [
        {
          "key": "<environment-variable-name>",
          "value": "<Path:$PATH>"
        }
      ],
      "deviceResources": [
        {
          "name": "<resource-name>",
          "path": "<resource-path>",
          "type": "device | volume"
        }
      ]
    },

 ********************************************************************************************
    "kernelConfigLocation": "",
    "greengrassLocation": "",
    "devices": [
      {
        "id": "<device-id>",
        "connectivity": {
          "protocol": "docker",
          "containerId": "<container-name | container-id>",
          "containerUser": "<user>"
        }
      }
    ]
  }
]
```

Semua bidang yang berisi nilai-nilai yang diperlukan seperti yang dijelaskan di sini:

`id`

ID alfanumerik yang ditetapkan pengguna secara unik mengidentifikasi kumpulan perangkat yang disebut kolam perangkat. Perangkat yang termasuk dalam suatu kolam harus memiliki perangkat

keras yang identik. Ketika Anda menjalankan serangkaian tes, perangkat di kolam tersebut digunakan untuk memparalelkan beban kerja. Beberapa perangkat digunakan untuk menjalankan tes yang berbeda.

`sku`

Nilai alfanumerik yang secara unik mengidentifikasi perangkat yang diuji. SKU digunakan untuk melacak forum berkualitas.

> **Note**
>
> Jika Anda ingin mencantumkan forum Anda di AWS Partner Katalog Perangkat, SKU yang Anda tentukan di sini harus sesuai dengan SKU yang Anda gunakan dalam proses daftar.

`features`

Array yang berisi fitur perangkat yang didukung. Semua fitur diperlukan.

`os` dan `arch`

Kombinasi sistem operasi (OS) dan arsitektur yang didukung:

- `linux`, `x86_64`
- `linux`, `armv6l`
- `linux`, `armv7l`
- `linux`, `aarch64`
- `ubuntu`, `x86_64`
- `openwrt`, `armv7l`
- `openwrt`, `aarch64`

> **Note**
>
> Jika anda menggunakan IDT untuk mengetes AWS IoT Greengrass berjalan dalam kontainer Docker, hanya arsitektur x86_64 Docker yang didukung.

`container`

Memvalidasi bahwa perangkat memenuhi semua persyaratan perangkat lunak dan perangkat keras untuk menjalankan fungsi Lambda dalam mode kontainer pada core Greengrass.

Nilai yang valid adalah `yes` atau `no`.

`docker`

Memvalidasi bahwa perangkat memenuhi semua dependensi teknis yang diperlukan untuk menggunakan konektor deployment aplikasi Greengrass Docker untuk menjalankan kontainer

Nilai yang valid adalah `yes` atau `no`.

`streamManagement`

Memvalidasi bahwa perangkat memenuhi semua dependensi teknis yang diperlukan untuk menjalankan AWS IoT Greengrass pengelola aliran.

Nilai yang valid adalah `yes` atau `no`.

`hsi`

Memverifikasi bahwa perpustakaan bersama HSI yang disediakan dapat antarmuka dengan modul keamanan perangkat keras (HSM) dan mengimplementasikan PKCS #11 API yang diperlukan dengan benar. HSM dan perpustakaan bersama harus dapat menandatangani CSR, melakukan operasi TLS, dan memberikan panjang kunci yang benar dan algoritma kunci publik.

Nilai yang valid adalah `yes` atau `no`.

`ml`

Memvalidasi bahwa perangkat memenuhi semua dependensi teknis yang diperlukan untuk melakukan inferensi ML secara lokal.

Nilai yang valid dapat berupa kombinasi `mxnet`, `tensorflow`, `dlr`, dan `no` (misalnya, `mxnet`, `mxnet,tensorflow`, `mxnet,tensorflow,dlr`, atau `no`).

`mlLambdaContainerizationMode`

Memvalidasi bahwa perangkat memenuhi semua dependensi teknis yang diperlukan untuk melakukan inferensi ML dalam mode kontainer pada perangkat Greengrass.

Nilai yang valid adalah `container`, `process`, atau `both`.

`processor`

Memvalidasi bahwa perangkat memenuhi semua persyaratan perangkat keras untuk jenis prosesor tertentu.

Nilai yang valid adalah `cpu` atau `gpu`.

> **Note**
>
> Jika Anda tidak ingin menggunakan `container`, `docker`, `streamManager`, `hsi`, atau `ml` fitur, Anda dapat mengatur sesuai `value` ke `no`.
> Docker hanya mendukung kualifikasi fitur untuk `streamManagement` dan `ml`.

`machineLearning`

Tidak wajib. Informasi konfigurasi untuk tes kualifikasi ML. Untuk informasi selengkapnya, lihat the section called “Konfigurasikan device.json untuk kualifikasi ML”.

`hsm`

Tidak wajib. Informasi konfigurasi untuk pengetesan dengan AWS IoT Greengrass Hardware Security Module (HSM). Jika tidak, `hsm` properti harus dihilangkan. Untuk informasi selengkapnya, lihat Integrasi keamanan perangkat keras.

Properti ini hanya berlaku jika `connectivity.protocol` diatur ke `ssh`.

`hsm.p11Provider`

Jalur absolut ke perpustakaan libdl-loadable implementasi PKCS #11.

`hsm.slotLabel`

Melabelkan slot yang digunakan untuk mengidentifikasi modul perangkat keras.

`hsm.slotUserPin`

PIN pengguna yang digunakan untuk mengotentikasi AWS IoT Greengrass core untuk modul.

`hsm.privateKeyLabel`

Label yang digunakan untuk mengidentifikasi kunci dalam modul perangkat keras.

`hsm.openSSLEngine`

Jalur absolut untuk OpenSSL file mesin `.so` yang memungkinkan dukungan PKCS #11 di OpenSSL. Digunakan oleh AWS IoT Greengrass agen update OTA.

`devices.id`

Pengenal unik yang ditetapkan pengguna untuk perangkat yang sedang dites.

`connectivity.protocol`

Protokol komunikasi yang digunakan untuk berkomunikasi dengan perangkat ini. Saat ini, satu-satunya nilai yang didukung adalah `ssh` untuk perangkat fisik dan `docker` untuk kontainer Docker.

`connectivity.ip`

Alamat IP perangkat yang sedang dites.

Properti ini hanya berlaku jika `connectivity.protocol` diatur ke `ssh`.

`connectivity.containerId`

ID kontainer atau nama kontainer Docker yang sedang diuji.

Properti ini hanya berlaku jika `connectivity.protocol` diatur ke `docker`.

`connectivity.auth`

Informasi autentikasi untuk koneksi tersebut.

Properti ini hanya berlaku jika `connectivity.protocol` diatur ke `ssh`.

`connectivity.auth.method`

Metode autentikasi yang digunakan untuk mengakses perangkat melalui protokol konektivitas yang diberikan.

Nilai yang didukung adalah:

- `pki`
- `password`

`connectivity.auth.credentials`

Kredensial yang digunakan untuk autentikasi.

`connectivity.auth.credentials.password`

Kata sandi yang digunakan untuk masuk ke perangkat yang sedang diuji.

Nilai ini hanya berlaku jika `connectivity.auth.method` diatur ke `password`.

`connectivity.auth.credentials.privKeyPath`

Jalur lengkap ke kunci privat yang digunakan untuk masuk ke perangkat yang sedang diuji.

Nilai ini hanya berlaku jika `connectivity.auth.method` diatur ke `pki`.

`connectivity.auth.credentials.user`

Nama pengguna untuk masuk ke perangkat yang sedang di.tes.

`connectivity.auth.credentials.privKeyPath`

Jalur lengkap ke kunci privat yang digunakan untuk masuk ke perangkat yang sedang dites.

`connectivity.port`

Tidak wajib. Jumlah port yang akan digunakan untuk koneksi SSH.

Nilai default adalah 22.

Properti ini berlaku hanya jika `connectivity.protocol` diatur ke `ssh`.

`greengrassLocation`

Lokasi AWS IoT Greengrass perangkat lunak Core pada perangkat Anda.

Untuk perangkat fisik, nilai ini hanya digunakan ketika Anda menggunakan instalasi yang ada AWS IoT Greengrass. Gunakan atribut ini untuk memberitahu IDT untuk menggunakan versi AWS IoT Greengrass perangkat lunak inti yang diinstal pada perangkat Anda.

Ketika menjalankan tes kontainer wadah Docker dari gambar Docker atau Dockerfile disediakan oleh AWS IoT Greengrass, tetapkan nilai ini ke `/greengrass`.

`kernelConfigLocation`

Tidak wajib. Jalur menuju file konfigurasi kernel. AWS IoT Tester Perangkat menggunakan file ini untuk memeriksa apakah perangkat memiliki fitur kernel yang diperlukan yang diaktifkan. Jika tidak ditentukan, IDT menggunakan jalur berikut untuk mencari file konfigurasi kernel: `/`

`proc/config.gz` dan `/boot/config-`*`<kernel-version>`*. AWS IoT Tester Perangkat menggunakan jalur pertama menemukan.

Konfigurasikan device.json untuk kualifikasi ML

Bagian ini menjelaskan properti opsional dalam file konfigurasi perangkat yang berlaku untuk kualifikasi ML. Jika Anda berencana untuk menjalankan tes untuk kualifikasi ML, Anda harus menentukan properti yang berlaku untuk kasus penggunaan Anda.

Anda dapat menggunakan `device-ml.json` templat untuk menentukan pengaturan konfigurasi untuk perangkat Anda. Templat ini berisi properti opsional ML. Anda juga dapat menggunakan `device.json` dan menambahkan properti kualifikasi ML. File ini terletak di *`<device-tester-extract-location>`*`/configs` dan termasuk properti kualifikasi ML. Jika Anda menggunakan `device-ml.json`, Anda harus mengubah nama file ke `device.json` sebelum Anda menjalankan tes IDT.

Untuk informasi tentang properti konfigurasi perangkat yang tidak berlaku untuk kualifikasi ML, lihat the section called “Konfigurasikan device.json”.

`ml` dalam `features` array

Kerangka kerja ML bahwa papan forum mendukung. Properti ini membutuhkan IDT v3.1.0 atau yang lebih baru.

- Jika forum Anda hanya mendukung satu kerangka kerja, tentukan kerangka kerjanya. Misalnya:

```
{
    "name": "ml",
    "value": "mxnet"
}
```

- Jika forum Anda mendukung beberapa kerangka kerja, tentukan kerangka kerja sebagai daftar yang dipisahkan koma. Misalnya:

```
{
    "name": "ml",
    "value": "mxnet,tensorflow"
}
```

`mlLambdaContainerizationMode` di `features` array

Pada Mode kontainerisasi yang ingin Anda tes. Properti ini membutuhkan IDT v3.1.0 atau yang lebih baru.

- Memilih `process` untuk menjalankan kode inferensi ML dengan fungsi Lambda non-containerized. Opsi ini memerlukan AWS IoT Greengrass v1.10.x atau yang lebih baru.
- Memilih `container` untuk menjalankan kode inferensi ML dengan fungsi Lambda kontainer.
- Memilih `both` untuk menjalankan kode inferensi ML dengan kedua mode. Opsi ini memerlukan AWS IoT Greengrass v1.10.x atau yang lebih baru.

`processor` di `features` array

Mengindikasikan akselerator perangkat keras yang didukung forum Anda. Properti ini membutuhkan IDT v3.1.0 atau yang lebih baru.

- Memilih `cpu` jika forum Anda menggunakan CPU sebagai prosesor.
- Memilih `gpu` jika forum Anda menggunakan GPU sebagai prosesor.

`machineLearning`

Tidak wajib. Informasi konfigurasi untuk tes kualifikasi ML. Properti ini membutuhkan IDT v3.1.0 atau yang lebih baru.

`dlrModelPath`

Diperlukan untuk menggunakan `dlr` kerangka kerja. Jalur absolut ke direktori model terkompilasi DLR Anda, yang harus diberi nama `resnet18`. Untuk informasi selengkapnya, lihat the section called “Kompilasi model DLR”.

> **Note**
>
> Berikut ini adalah contoh jalur di macOS: `/Users/<user>/Downloads/resnet18`.

`environmentVariables`

Array pasangan kunci-nilai yang dinamis dapat lulus pengaturan untuk tes inferensi ML. Opsional untuk perangkat CPU. Anda dapat menggunakan bagian ini untuk menambahkan variabel lingkungan khusus kerangka kerja yang diperlukan oleh jenis perangkat Anda. Untuk informasi tentang persyaratan ini, lihat situs resmi kerangka kerja atau perangkat. Sebagai contoh, untuk menjalankan tes inferensi MXNet pada beberapa perangkat, variabel lingkungan berikut mungkin diperlukan.

```
"environmentVariables": [
    ...
    {
        "key": "PYTHONPATH",
        "value": "$MXNET_HOME/python:$PYTHONPATH"
    },
    {
        "key": "MXNET_HOME",
        "value": "$HOME/mxnet/"
    },
    ...
]
```

> **Note**
>
> Bidang `value` mungkin bervariasi berdasarkan instalasi MXNet Anda.

Jika anda sedang mengetes fungsi Lambda yang berjalan dengan kontainerisasi pada perangkat GPU, tambahkan variabel lingkungan untuk perpustakaan GPU. Hal ini memungkinkan GPU untuk melakukan komputasi. Untuk menggunakan perpustakaan GPU yang berbeda, lihat dokumentasi resmi untuk perpustakaan atau perangkat.

> **Note**
>
> Konfigurasikan kunci berikut jika `mlLambdaContainerizationMode` fitur diatur ke `container` atau `both`.

```
"environmentVariables": [
    {
        "key": "PATH",
        "value": "<path/to/software/bin>:$PATH"
    },
    {
        "key": "LD_LIBRARY_PATH",
        "value": "<path/to/ld/lib>"
    },
    ...
]
```

`deviceResources`

Diperlukan oleh perangkat GPU. Berisi sumber daya lokal yang dapat diakses oleh fungsi Lambda. Gunakan bagian ini untuk menambahkan perangkat lokal dan volume sumber daya.

- Untuk sumber daya perangkat, tentukan `"type": "device"`. Untuk perangkat GPU, sumber daya perangkat harus berupa file perangkat terkait GPU di bawah `/dev`.

> **Note**
>
> Direktori `/dev/shm` adalah pengecualian. Hal ini dapat dikonfigurasi sebagai sumber daya volume saja.

- Untuk sumber daya volume, tentukan `"type": "volume"`.

Jalankan AWS IoT Greengrass suite kualifikasi

Setelah Anda mengatur konfigurasi yang diperlukan, Anda bisa memulai tes. Waktu aktif dari rangkaian uji penuh tergantung pada perangkat keras Anda. Untuk referensi, memakan waktu sekitar 30 menit untuk menyelesaikan tes suite penuh pada Raspberry Pi 3B.

Contoh berikut `run-suite` menunjukkan cara menjalankan tes kualifikasi untuk kolam perangkat. Kolam perangkat adalah satu set perangkat yang sama.

IDT v3.0.0 and later

Jalankan semua grup tes dalam tes suite tertentu.

```
devicetester_[linux | mac | win_x86-64] run-suite --suite-id GGQ_1.0.0 --pool-
id <pool-id>
```

Gunakan `list-suites` perintah untuk daftar tes suite yang ada di `tests` folder.

Jalankan grup tes tertentu dalam tes suite.

```
devicetester_[linux | mac | win_x86-64] run-suite --suite-id GGQ_1.0.0 --group-
id <group-id> --pool-id <pool-id>
```

Gunakan `list-groups` perintah untuk daftar grup tes di tes suite.

Jalankan kasus tes tertentu dalam grup tes.

```
devicetester_[linux | mac | win_x86-64] run-suite --group-id <group-id> --test-
id <test-id>
```

Jalankan beberapa tes kasus dalam grup tes.

```
devicetester_[linux | mac | win_x86-64] run-suite --group-id <group-id> --test-
id <test-id1>,<test-id2>
```

Daftar tes kasus dalam grup tes.

```
devicetester_[linux | mac | win_x86-64] list-test-cases --group-id <group-id>
```

Opsi untuk `run-suite` perintah adalah opsional. Sebagai contoh, Anda dapat menghilangkan `pool-id` jika Anda hanya memiliki satu kolam perangkat yang didefinisikan di file `device.json` Anda. Atau, Anda bisa menghilangkan `suite-id` jika Anda ingin menjalankan folder `tests` versi terbaru.

> **Note**
>
> IDT meminta Anda jika versi rangkaian tes yang lebih baru tersedia secara online. Untuk informasi selengkapnya, lihat the section called “Atur perilaku update default”.

Untuk informasi lebih lanjut tentang `run-suite` dan perintah IDT lainnya, lihat the section called “Perintah IDT”.

IDT v2.3.0 and earlier

Jalankan semua grup tes di suite tertentu.

```
devicetester_[linux | mac | win_x86-64] run-suite --suite-id GGQ_1 --pool-
id <pool-id>
```

Jalankan grup tes tertentu.

```
devicetester_[linux | mac | win_x86-64] run-suite --suite-id GGQ_1 --group-
id <group-id> --pool-id <pool-id>
```

`suite-id` dan `pool-id` opsional jika Anda menjalankan tes suite tunggal pada kolam perangkat tunggal. Ini berarti bahwa Anda hanya memiliki satu perangkat kolam didefinisikan dalam file `device.json` Anda.

Periksa depedensi Greengrass

Kami merekomendasikan Anda menjalankan grup tes pemeriksa dependensi untuk memastikan semua dependensi Greengrass diinstal sebelum Anda menjalankan grup tes terkait. Misalnya:

- Jalankan `ggcdependencies` sebelum menjalankan grup tes kualifikasi core.
- Jalankan `containerdependencies` sebelum menjalankan grup tes kontainer-spesifik.
- Jalankan `dockerdependencies` sebelum menjalankan grup tes Docker-spesifik.
- Jalankan `ggcstreammanagementdependencies` sebelum menjalankan grup tes khusus pengelola aliran.

Atur perilaku update default

Ketika Anda memulai tes run, IDT memeriksa secara online untuk versi tes suite yang lebih baru. Jika tersedia, IDT meminta Anda untuk update ke versi terbaru yang tersedia. Anda dapat mengatur `upgrade-test-suite` (atau u) bendera untuk mengontrol perilaku update default. Nilai yang benar adalah:

- y. IDT mengunduh dan menggunakan versi terbaru yang tersedia.
- n (default). IDT menggunakan versi yang ditentukan dalam `suite-id` opsi. Jika `suite-id` tidak ditentukan, IDT menggunakan versi terbaru dalam folder `tests` ini.

Jika Anda tidak menyertakan `upgrade-test-suite` bendera, IDT meminta Anda ketika update tersedia dan menunggu 30 detik untuk input Anda (y atau n). Jika tidak ada input yang dimasukkan, defaultnya ke n dan terus menjalankan tes.

Contoh-contoh berikut menunjukkan kasus penggunaan umum untuk fitur ini:

Secara otomatis menggunakan tes terbaru yang tersedia untuk grup tes.

```
devicetester_linux run-suite -u y --group-id mqtt --pool-id DevicePool1
```

Jalankan tes dalam versi tes suite tertentu.

```
devicetester_linux run-suite -u n --suite-id GGQ_1.0.0 --group-id mqtt --pool-id
 DevicePool1
```

Prompt untuk update pada saat waktu aktif.

```
devicetester_linux run-suite --pool-id DevicePool1
```

IDT untuk AWS IoT Greengrass perintah

Perintah IDT terletak di *<device-tester-extract-location>*/bin direktori. Gunakan mereka untuk operasi berikut:

IDT v3.0.0 and later

`help`

Daftar informasi tentang perintah yang ditentukan.

`list-groups`

Mendaftar grup dalam rangkaian tes yang diberikan.

`list-suites`

Mencantumkan rangkaian tes yang tersedia.

`list-supported-products`

Mencantumkan produk yang didukung, dalam hal ini versi AWS IoT Greengrass, dan versi rangkaian uji untuk versi IDT saat ini.

`list-test-cases`

Mencantumkan uji kasus dalam grup uji yang diberikan. Opsi berikut didukung:

- `group-id`. Grup uji yang harus dicari. Opsi ini diperlukan dan harus menentukan satu grup.

`run-suite`

Menjalankan serangkaian tes pada kolam perangkat. Berikut ini adalah beberapa opsi yang didukung:

- `suite-id`. Versi rangkaian tes yang akan jalankan. Jika tidak ditentukan, IDT akan menggunakan versi terbaru dalam folder `tests`.
- `group-id`. Grup uji yang akan jalankan, sebagai daftar yang dipisahkan koma. Jika tidak ditentukan, IDT akan menjalankan semua grup uji di rangkaian tes.
- `test-id`. Uji kasus yang akan dijalankan, sebagai daftar yang dipisahkan koma. Ketika ditentukan, `group-id` harus menentukan satu grup.
- `pool-id`. Kolam perangkat yang akan diuji. Anda harus menentukan kolam jika Anda memiliki beberapa perangkat kolam yang didefinisikan dalam file `device.json` Anda.
- `upgrade-test-suite`. Mengontrol bagaimana update versi tes suite ditangani. Mulai IDT v3.0.0, IDT memeriksa secara online untuk memperbarui versi rangkaian pengujian. Untuk informasi selengkapnya, lihat the section called “Versi rangkaian pengujian”.
- `stop-on-first-failure`. Mengkonfigurasi IDT untuk menghentikan pelaksanaan pada kegagalan pertama. Pilihan ini harus digunakan dengan `group-id` untuk men-debug grup tes yang ditentukan. Jangan gunakan opsi ini ketika menjalankan tes suite penuh untuk menghasilkan laporan kualifikasi.
- `update-idt`. Mengatur respon untuk prompt untuk update IDT. `Y` ketika input menghentikan pelaksanaan tes jika IDT mendeteksi ada versi yang lebih baru. `N` ketika input terus melaksanakan tes.
- `update-managed-policy`. `Y` ketika input berhenti melaksanakan tes jika IDT mendeteksi bahwa kebijakan dikelola pengguna tidak diperbarui. `N` ketika input terus melaksanakan tes.

Untuk informasi lebih lanjut tentang `run-suite` opsi, gunakan `help` Opsi:

```
devicetester_[linux | mac | win_x86-64] run-suite -h
```

IDT v2.3.0 and earlier

`help`

Daftar informasi tentang perintah yang ditentukan.

`list-groups`

Mendaftar grup dalam rangkaian tes yang diberikan.

`list-suites`

Daftar tes suite yang tersedia.

`run-suite`

Menjalankan serangkaian tes pada kolam perangkat.

Untuk informasi lebih lanjut tentang `run-suite` opsi, gunakan `help` opsi:

```
devicetester_[linux | mac | win_x86-64] run-suite -h
```

Memahami hasil dan mencatat

Bagian ini menjelaskan bagaimana melihat dan menafsirkan laporan hasil IDT dan mencatat.

Melihat Hasil

Saat berjalan, IDT menuliskan kesalahan ke konsol, file log, dan laporan tes. Setelah IDT menyelesaikan tes suite kualifikasi, menghasilkan dua laporan tes. Laporan-laporan ini dapat ditemukan di *<device-tester-extract-location>*`/results/`*<execution-id>*`/`. Kedua laporan menangkap hasil dari pelaksanaan kualifikasi tes suite.

Ini `awsiotdevicetester_report.xml` adalah laporan tes kualifikasi yang Anda kirimkan ke AWS untuk mencantumkan perangkat Anda di AWS Partner Katalog Perangkat. Laporan tersebut berisi elemen berikut:

- Versi IDT.
- Versi AWS IoT Greengrass yang diuji.
- SKU dan nama kolam perangkat yang ditentukan dalam file `device.json`.
- Fitur kolam perangkat yang ditentukan dalam file `device.json`.
- Ringkasan agregat hasil tes.
- Perincian hasil tes oleh perpustakaan yang dites berdasarkan fitur perangkat (sebagai contoh, akses sumber daya lokal, bayangan, MQTT, dan sebagainya).

Laporan `GGQ_Result.xml` ada dalam format JUnit XML. Anda dapat mengintegrasikannya ke dalam integrasi berkelanjutan dan platform deployment seperti Jenkins, Bamboo, dan sebagainya. Laporan tersebut berisi elemen berikut:

- Ringkasan agregat hasil pengujian.
- Rincian hasil tes oleh AWS IoT Greengrass fungsi yang dites.

Menafsirkan laporan IDT

Bagian laporan di `awsiotdevicetester_report.xml` atau `awsiotdevicetester_report.xml` daftar tes yang dijalankan dan hasilnya.

Menandai XML pertama `<testsuites>` berisi ringkasan pelaksanaan tes. Misalnya:

```
<testsuites name="GGQ results" time="2299" tests="28" failures="0" errors="0"
 disabled="0">
```

Atribut yang digunakan dalam tanda **`<testsuites>`**

`name`

Nama dari tes suite.

`time`

Waktu, dalam hitungan detik, yang dibutuhkan untuk menjalankan suite kualifikasi.

`tests`

Jumlah tes yang dilaksanakan.

`failures`

Jumlah tes yang dijalankan, tetapi tidak lulus.

`errors`

Jumlah tes yang tidak dapat dilaksanakan oelh IDT.

`disabled`

Atribut ini tidak digunakan dan dapat diabaikan.

File `awsiotdevicetester_report.xml` berisi sebuah tanda `<awsproduct>` yang berisi informasi tentang produk yang sedang dites dan fitur produk yang divalidasi setelah menjalankan tes suite.

Atribut yang digunakan dalam tanda **`<awsproduct>`**

`name`

Nama produk yang sedang diuji.

`version`

Versi produk yang sedang diuji.

`features`

Fitur divalidasi. Fitur yang ditandai sebagai `required` wajib mengirimkan forum Anda untuk kualifikasi. Potongan berikut menunjukkan bagaimana informasi ini muncul di `awsiotdevicetester_report.xml` file.

```
<feature name="aws-iot-greengrass-no-container" value="supported" type="required"></
feature>
```

Fitur yang ditandai sebagai `optional` tidak diperlukan untuk kualifikasi. Potongan berikut menunjukkan fitur opsional.

```
<feature name="aws-iot-greengrass-container" value="supported" type="optional"></
feature>

<feature name="aws-iot-greengrass-hsi" value="not-supported" type="optional"></
feature>
```

Jika tidak ada kegagalan tes atau kesalahan untuk fitur yang diperlukan, perangkat Anda memenuhi persyaratan teknis untuk menjalankan AWS IoT Greengrass dan dapat bekerja sama dengan AWS IoT layanan. Jika Anda ingin mencantumkan perangkat Anda di Katalog Perangkat AWS Partner, Anda dapat menggunakan laporan ini sebagai bukti kualifikasi.

Jika terjadi kegagalan atau kesalahan uji, Anda dapat mengidentifikasi pengujian yang gagal tersebut dengan meninjau tanda XML `<testsuites>`. Tag XML `<testsuite>` di dalam tag `<testsuites>` menunjukkan ringkasan hasil tes untuk grup uji. Misalnya:

```
<testsuite name="combination" package="" tests="1" failures="0" time="161" disabled="0"
 errors="0" skipped="0">
```

Format ini serupa dengan tanda `<testsuites>`, tetapi dengan atribut `skipped` yang tidak digunakan dan dapat diabaikan. Di dalam masing-masing tanda XML `<testsuite>`, terdapat tanda `<testcase>` untuk setiap tes yang dieksekusi untuk suatu grup uji. Misalnya:

```
<testcase classname="Security Combination (IPD + DCM) Test Context" name="Security
 Combination IP Change Tests sec4_test_1: Should rotate server cert when IPD disabled
```

```
 and following changes are made:Add CIS conn info and Add another CIS conn info"
 attempts="1"></testcase>>
```

Atribut yang digunakan dalam tanda **<testcase>**

`name`

Nama tes.

`attempts`

Berapa kali IDT mengeksekusi uji kasus.

Ketika tes gagal atau kesalahan terjadi, tag `<failure>` atau `<error>` akan ditambahkan ke tag `<testcase>` dengan informasi untuk pemecahan masalah. Misalnya:

```
<testcase classname="mcu.Full_MQTT" name="AFQP_MQTT_Connect_HappyCase" attempts="1">
 <failure type="Failure">Reason for the test failure</failure>
 <error>Reason for the test execution error</error>
</testcase>
```

Melihat log

IDT menghasilkan log dari pelaksanaan tes di *<devicetester-extract-location>*`/results/`*<execution-id>*`/logs`. Dua set log dihasilkan:

`test_manager.log`

Log yang dihasilkan dari komponen Test Manager AWS IoT Tester Perangkat (sebagai contoh, log yang terkait dengan konfigurasi, pengurutan tes, dan pembuatan laporan).

<test_case_id>`.log (for example, ota.log)`

Log dari grup tes, termasuk log dari perangkat yang dites. Ketika tes gagal, file tar.gz yang berisi log perangkat yang dites untuk tes dibuat (sebagai contoh, `ota_prod_test_1_ggc_logs.tar.gz`).

Untuk informasi selengkapnya, lihat IDT untuk AWS IoT Greengrass pemecahan masalah.

Gunakan IDT untuk mengembangkan dan menjalankan rangkaian pengujian Anda sendiri

Dimulai di IDT v4.0.0, IDT untuk AWS IoT Greengrass menggabungkan pengaturan konfigurasi standar dan format hasil dengan lingkungan rangkaian percobaan yang memungkinkan Anda untuk mengembangkan rangkaian pengujian khusus untuk perangkat dan perangkat lunak perangkat Anda. Anda dapat menambahkan pengujian khusus untuk validasi internal Anda sendiri atau memberikannya kepada pelanggan Anda untuk verifikasi perangkat.

Gunakan IDT untuk mengembangkan dan menjalankan rangkaian tes kustom, sebagai berikut:

Untuk mengembangkan rangkaian tes kustom

- Buat rangkaian test dengan logika tes kustom untuk perangkat Greengrass yang ingin Anda uji.
- Sediakan IDT dengan rangkaian tes kustom Anda untuk menguji runner. Sertakan informasi tentang konfigurasi pengaturan khusus untuk rangkaian pengujian Anda.

Untuk menjalankan rangkaian uji kustom

- Atur perangkat yang ingin Anda uji.
- Terapkan konfigurasi pengaturan yang diperlukan oleh rangkaian tes yang ingin Anda gunakan.
- Gunakan IDT untuk menjalankan rangkaian tes kustom Anda.
- Lihat hasil tes dan log eksekusi untuk tes yang dijalankan oleh IDT.

Unduh versi terbaru dari AWS IoT Penguji Perangkat untuk AWS IoT Greengrass

Unduh versi terbaru IDT dan mengekstraksi perangkat lunak ke lokasi pada sistem file Anda di mana Anda telah membaca dan menulis izin.

> **Note**
>
> IDT tidak mendukung yang sedang dijalankan oleh beberapa pengguna dari lokasi bersama, seperti direktori NFS atau folder bersama jaringan Windows. Kami sarankan Anda mengekstraksi paket IDT ke drive lokal dan menjalankan biner IDT pada workstation lokal Anda.

Windows memiliki batasan panjang jalur 260 karakter. Jika Anda menggunakan Windows, ekstraksi IDT ke direktori root seperti `C:\` atau `D:\` agar jalur Anda tetap di bawah batas 260 karakter.

Alur kerja pembuatan rangkaian uji

Rangkaian uji terdiri dari tiga jenis file:

- File konfigurasi JSON yang menyediakan IDT dengan informasi tentang cara menjalankan rangkaiain tes.
- File yang dapat dieksekusi yang digunakan oleh IDT untuk menjalankan uji kasus.
- File tambahan diperlukan untuk menjalankan tes.

Selesaikan langkah-langkah dasar berikut untuk membuat tes IDT kustom:

1. Buat file konfigurasi JSON untuk rangkaian tes Anda.
2. Buat uji kasus yang dapat dieksekusi yang berisi logika ujian untuk rangkaian tes anda.
3. Verifikasi dan dokumentasi informasi konfigurasi yang diperlukan untuk menguji runner untuk menjalankan rangkaian tes.
4. Verifikasi bahwa IDT dapat menjalankan rangkain tes Anda dan buat hasil pengujian seperti yang diharapkan.

Untuk cepat membangun rangkaian kustom sampel dan menjalankannya, ikuti petunjuk di Tutorial: Bangun sampel rangkaian tes IDT.

Untuk memulai membuat rangkaian tes kustom di Python, lihat Tutorial: Kembangkan rangkaian tes IDT sederhana.

Tutorial: Bangun sampel rangkaian tes IDT

Unduhan AWS IoT Penguji Perangkat mencakup kode sumber untuk rangkaian pengujian sampel. Anda dapat menyelesaikan tutorial ini untuk membangun dan menjalankan rangkaian pengujian sampel untuk memahami bagaimana Anda dapat menggunakan AWS IoT Penguji Perangkat untuk AWS IoT Greengrass untuk menjalankan rangkaian pengujian khusus.

Dalam tutorial ini, Anda akan menyelesaikan langkah-langkah berikut:

1. Bangun rangkaian uji sampel
2. Gunakan IDT untuk menjalankan rangkaian uji sampel

Prasyarat

Untuk menyelesaikan tutorial ini, Anda memerlukan hal berikut ini:

- Persyaratan komputer host
  - Versi terbaru dari AWS IoT Penguji Perangkat
  - Python 3.7 atau yang lebih baru

    Untuk memeriksa versi Python yang diinstal pada komputer Anda, jalankan perintah berikut:

    ```
    python3 --version
    ```

    Pada Windows, jika penggunaan perintah ini menghasilkan kesalahan, gunakan `python --version` sebagai gantinya. Jika nomor versi yang dikembalikan adalah 3,7 atau lebih besar, jalankan perintah berikut di terminal Powershell untuk mengatur `python3` sebagai alias untuk perintah `python`.

    ```
    Set-Alias -Name "python3" -Value "python"
    ```

    Jika tidak ada informasi versi yang dikembalikan atau jika nomor versi kurang dari 3,7, ikuti petunjuk di Mengunduh Py untuk menginstal Python 3.7+. Untuk informasi selengkapnya, lihat dokumentasi Python.
  - urllib3

    Untuk memverifikasi bahwa `urllib3` diinstal dengan benar, jalankan perintah berikut:

    ```
    python3 -c 'import urllib3'
    ```

    Jika `urllib3` belum terinstal, gunakan perintah berikut untuk menginstalnya:

    ```
    python3 -m pip install urllib3
    ```

- Persyaratan perangkat
  - Perangkat dengan sistem operasi Linux dan koneksi jaringan ke jaringan yang sama dengan komputer host Anda.

    Kami menyarankan agar Anda menggunakan Raspberry Pi dengan OS Raspberry Pi. Pastikan Anda mengatur SSH pada Raspberry Pi Anda untuk terhubung secara jarak jauh ke sana.

Konfigurasi informasi perangkat untuk IDT

Konfigurasi informasi perangkat Anda untuk IDT untuk menjalankan tes. Anda harus memperbarui templat `device.json` yang terletak di folder *`<device-tester-extract-location>`*`/configs` dengan informasi berikut.

```
[
  {
    "id": "pool",
    "sku": "N/A",
    "devices": [
      {
        "id": "<device-id>",
        "connectivity": {
          "protocol": "ssh",
          "ip": "<ip-address>",
          "port": "<port>",
          "auth": {
            "method": "pki | password",
            "credentials": {
              "user": "<user-name>",
              "privKeyPath": "/path/to/private/key",
              "password": "<password>"
            }
          }
        }
      }
    ]
  }
]
```

Di objek `devices`, berikan informasi berikut:

`id`

Pengenal unik yang ditetapkan pengguna untuk perangkat Anda.

`connectivity.ip`

Alamat IP perangkat Anda.

`connectivity.port`

Tidak wajib. Nomor port yang digunakan untuk koneksi SSH ke perangkat Anda.

`connectivity.auth`

Informasi autentikasi untuk koneksi tersebut.

Properti ini hanya berlaku jika `connectivity.protocol` diatur ke `ssh`.

`connectivity.auth.method`

Metode autentikasi yang digunakan untuk mengakses perangkat melalui protokol konektivitas yang diberikan.

Nilai yang didukung adalah:

- `pki`
- `password`

`connectivity.auth.credentials`

Kredensial yang digunakan untuk autentikasi.

`connectivity.auth.credentials.user`

Nama pengguna yang digunakan untuk masuk ke perangkat Anda.

`connectivity.auth.credentials.privKeyPath`

Jalur lengkap ke kunci pribadi yang digunakan untuk masuk ke perangkat Anda.

Nilai ini hanya berlaku jika `connectivity.auth.method` diatur ke `pki`.

`devices.connectivity.auth.credentials.password`

Kata sandi yang digunakan untuk masuk ke perangkat Anda.

Nilai ini hanya berlaku jika `connectivity.auth.method` diatur ke `password`.

> **Note**
>
> Tentukan `privKeyPath` hanya jika `method` diatur ke `pki`.
> Tentukan `password` hanya jika `method` diatur ke `password`.

Bangun rangkaian uji sampel

Folder *<device-tester-extract-location>*`/samples/python` berisi contoh file konfigurasi, kode sumber, dan SDK Klien IDT yang dapat Anda gabungkan ke dalam rangkaian tes dengan menggunakan skrip build yang disediakan. Pohon direktori berikut menunjukkan lokasi file contoh ini:

```
<device-tester-extract-location>
### ...
### tests
### samples
#   ### ...
#   ### python
#       ### configuration
#       ### src
#       ### build-scripts
#           ### build.sh
#           ### build.ps1
### sdks
    ### ...
    ### python
        ### idt_client
```

Untuk membangun rangkaian tes, jalankan perintah berikut pada komputer host Anda:

Windows

```
cd <device-tester-extract-location>/samples/python/build-scripts
./build.ps1
```

Linux, macOS, or UNIX

```
cd <device-tester-extract-location>/samples/python/build-scripts
```

```
./build.sh
```

Hal ini akan menciptakan rangkaian uji sampel di folder `IDTSampleSuitePython_1.0.0` dalam folder *<device-tester-extract-location>*`/tests`. Tinjau file di folder `IDTSampleSuitePython_1.0.0` untuk memahami bagaimana sampel rangkaian tes dibangun dan untuk melihat berbagai contoh uji kasus executable dan konfigurasi tes file JSON.

Langkah berikutnya: Gunakan IDTjalankan rangkaian uji sampelyang Anda ciptakan.

Gunakan IDT untuk menjalankan rangkaian uji sampel

Untuk membangun rangkaian tes, jalankan perintah berikut pada komputer host Anda:

```
cd <device-tester-extract-location>/bin
./devicetester_[linux | mac | win_x86-64] run-suite --suite-id IDTSampleSuitePython
```

IDT menjalankan sampel rangkaian tes dan mengalirkan hasil ke konsol. Ketika tes telah selesai berjalan, Anda akan melihat informasi berikut:

```
========== Test Summary ==========
Execution Time:         5s
Tests Completed:        4
Tests Passed:           4
Tests Failed:           0
Tests Skipped:          0
---------------------------------
Test Groups:
    sample_group:       PASSED
---------------------------------
Path to IoT Device Tester Report: /path/to/devicetester/
results/87e673c6-1226-11eb-9269-8c8590419f30/awsiotdevicetester_report.xml
Path to Test Execution Logs: /path/to/devicetester/
results/87e673c6-1226-11eb-9269-8c8590419f30/logs
Path to Aggregated JUnit Report: /path/to/devicetester/
results/87e673c6-1226-11eb-9269-8c8590419f30/IDTSampleSuitePython_Report.xml
```

Pemecahan Masalah

Gunakan informasi berikut untuk membantu menyelesaikan masalah dengan menyelesaikan tutorial.

Uji kasus tidak berjalan dengan sukses

Jika tes tidak berjalan sukses, IDT akan mengalirkan log kesalahan ke konsol yang dapat membantu Anda memecahkan masalah uji coba. Pastikan Anda memenuhi semua prasyarat untuk tutorial ini.

Tidak dapat menyambung ke perangkat yang sedang diuji

Verifikasi hal berikut:

- File `device.json` Anda berisi alamat IP, port, dan informasi autentikasi yang benar.
- Anda dapat terhubung ke perangkat Anda melalui SSH dari komputer host Anda.

Tutorial: Kembangkan rangkaian tes IDT sederhana

Sebuah rangkaian pengujian menggabungkan berikut:

- Executable tes yang berisi logika tes
- File konfigurasi JSON yang menggambarkan rangkaian tes

Tutorial ini menunjukkan cara menggunakan IDT untuk AWS IoT Greengrass untuk mengembangkan rangkaian tes Python yang berisi kasus tes tunggal. Dalam tutorial ini, Anda akan melakukan langkah-langkah berikut:

1. Buat direktori rangkaian tes
2. Buat file konfigurasi JSON
3. Buat executable uji kasus
4. Jalankan rangkaian tes

Prasyarat

Untuk menyelesaikan tutorial ini, Anda memerlukan hal berikut ini:

- Persyaratan komputer host
  - Versi terbaru dari AWS IoT Penguji Perangkat
  - Python 3.7 atau yang lebih baru

    Untuk memeriksa versi Python yang diinstal pada komputer Anda, jalankan perintah berikut:

    ```
    python3 --version
    ```

Pada Windows, jika penggunaan perintah ini menghasilkan kesalahan, gunakan `python --version` sebagai gantinya. Jika nomor versi yang dikembalikan adalah 3,7 atau lebih besar, jalankan perintah berikut di terminal Powershell untuk mengatur `python3` sebagai alias untuk perintah `python`.

```
Set-Alias -Name "python3" -Value "python"
```

Jika tidak ada informasi versi yang dikembalikan atau jika nomor versi kurang dari 3,7, ikuti petunjuk di Mengunduh Py untuk menginstal Python 3.7+. Untuk informasi selengkapnya, lihat dokumentasi Python.

- urllib3

  Untuk memverifikasi bahwa `urllib3` diinstal dengan benar, jalankan perintah berikut:

  ```
  python3 -c 'import urllib3'
  ```

  Jika `urllib3` belum terinstal, gunakan perintah berikut untuk menginstalnya:

  ```
  python3 -m pip install urllib3
  ```

- Persyaratan perangkat
  - Perangkat dengan sistem operasi Linux dan koneksi jaringan ke jaringan yang sama dengan komputer host Anda.

    Kami menyarankan agar Anda menggunakan Raspberry Pi dengan OS Raspberry Pi. Pastikan Anda mengatur SSH pada Raspberry Pi Anda untuk terhubung secara jarak jauh ke sana.

Buat direktori rangkaian tes

IDT secara logis memisahkan uji kasus ke dalam grup uji dalam setiap rangkaian tes. Setiap uji kasus harus berada di dalam grup uji. Untuk tutorial ini, buat folder bernama `MyTestSuite_1.0.0` dan buat pohon direktori berikut dalam folder ini:

```
MyTestSuite_1.0.0
### suite
    ### myTestGroup
        ### myTestCase
```

Buat file konfigurasi JSON

Rangkaian tes Anda harus berisi file konfigurasi JSON yang diperlukan berikut ini:

File JSON yang dibutuhkan

`suite.json`

Berisi informasi tentang rangkaian pengujian. Lihat Konfigurasikan suite.json.

`group.json`

Berisi informasi tentang grup uji. Anda harus membuat file `group.json` untuk setiap grup uji di rangkaian tes Anda. Lihat Konfigurasikan group.json.

`test.json`

Berisi informasi tentang grup uji. Anda harus membuat file `test.json` untuk setiap grup uji di rangkaian tes Anda. Lihat Konfigurasikan test.json.

1. Di folder `MyTestSuite_1.0.0/suite`, buat file `suite.json` dengan struktur berikut:

```
{
    "id": "MyTestSuite_1.0.0",
    "title": "My Test Suite",
    "details": "This is my test suite.",
    "userDataRequired": false
}
```

2. Di folder `MyTestSuite_1.0.0/myTestGroup`, buat file `group.json` dengan struktur berikut:

```
{
    "id": "MyTestGroup",
    "title": "My Test Group",
    "details": "This is my test group.",
    "optional": false
}
```

3. Di folder `MyTestSuite_1.0.0/myTestGroup/myTestCase`, buat file `test.json` dengan struktur berikut:

```
{
```

```
    "id": "MyTestCase",
    "title": "My Test Case",
    "details": "This is my test case.",
    "execution": {
        "timeout": 300000,
        "linux": {
            "cmd": "python3",
            "args": [
                "myTestCase.py"
            ]
        },
        "mac": {
            "cmd": "python3",
            "args": [
                "myTestCase.py"
            ]
        },
        "win": {
            "cmd": "python3",
            "args": [
                "myTestCase.py"
            ]
        }
    }
}
```

Pohon direktori untuk folder `MyTestSuite_1.0.0` Anda sekarang akan terlihat seperti berikut ini:

```
MyTestSuite_1.0.0
### suite
    ### suite.json
    ### myTestGroup
        ### group.json
        ### myTestCase
            ### test.json
```

Dapatkan SDK klien IDT

Anda menggunakan SDK Klien IDT untuk memungkinkan IDT berinteraksi dengan perangkat yang sedang diuji dan melaporkan hasil pengujian. Untuk tutorial ini, Anda akan menggunakan versi Python dari SDK.

Dari folder *<device-tester-extract-location>*/sdks/python/, salin folder idt_client ke folder MyTestSuite_1.0.0/suite/myTestGroup/myTestCase Anda.

Untuk memverifikasi bahwa SDK berhasil disalin, jalankan perintah berikut.

```
cd MyTestSuite_1.0.0/suite/myTestGroup/myTestCase
python3 -c 'import idt_client'
```

Buat executable uji kasus

Executable uji kasus berisi logika tes yang ingin Anda jalankan. Sebuah rangkaian tes dapat berisi beberapa executable uji kasus. Untuk tutorial ini, Anda hanya akan membuat satu executable uji kasus.

1. Buat file rangkaian test.

   Di folder MyTestSuite_1.0.0/suite/myTestGroup/myTestCase, buat file myTestCase.py dengan konten berikut:

   ```
   from idt_client import *

   def main():
       # Use the client SDK to communicate with IDT
       client = Client()

   if __name__ == "__main__":
       main()
   ```

2. Gunakan fungsi SDK klien untuk menambahkan logika uji berikut ke file myTestCase.py Anda:

   a. Jalankan perintah SSH pada perangkat yang diuji.

      ```
      from idt_client import *

      def main():
          # Use the client SDK to communicate with IDT
          client = Client()

          # Create an execute on device request
          exec_req = ExecuteOnDeviceRequest(ExecuteOnDeviceCommand("echo 'hello
       world'"))
      ```

```
    # Run the command
    exec_resp = client.execute_on_device(exec_req)

    # Print the standard output
    print(exec_resp.stdout)

if __name__ == "__main__":
    main()
```

b. Kirim hasil tes ke IDT.

```
from idt_client import *

def main():
    # Use the client SDK to communicate with IDT
    client = Client()

    # Create an execute on device request
    exec_req = ExecuteOnDeviceRequest(ExecuteOnDeviceCommand("echo 'hello
 world'"))

    # Run the command
    exec_resp = client.execute_on_device(exec_req)

    # Print the standard output
    print(exec_resp.stdout)

    # Create a send result request
    sr_req = SendResultRequest(TestResult(passed=True))

    # Send the result
    client.send_result(sr_req)

if __name__ == "__main__":
    main()
```

Konfigurasi informasi perangkat untuk IDT

Konfigurasi informasi perangkat Anda untuk IDT untuk menjalankan tes. Anda harus memperbarui templat `device.json` yang terletak di folder *<device-tester-extract-location>*`/configs` dengan informasi berikut.

```
[
  {
    "id": "pool",
    "sku": "N/A",
    "devices": [
      {
        "id": "<device-id>",
        "connectivity": {
          "protocol": "ssh",
          "ip": "<ip-address>",
          "port": "<port>",
          "auth": {
            "method": "pki | password",
            "credentials": {
              "user": "<user-name>",
              "privKeyPath": "/path/to/private/key",
              "password": "<password>"
            }
          }
        }
      }
    ]
  }
]
```

Di objek `devices`, berikan informasi berikut:

`id`

Pengenal unik yang ditetapkan pengguna untuk perangkat Anda.

`connectivity.ip`

Alamat IP perangkat Anda.

`connectivity.port`

Tidak wajib. Nomor port yang digunakan untuk koneksi SSH ke perangkat Anda.

`connectivity.auth`

Informasi autentikasi untuk koneksi tersebut.

Properti ini hanya berlaku jika `connectivity.protocol` diatur ke `ssh`.

`connectivity.auth.method`

Metode autentikasi yang digunakan untuk mengakses perangkat melalui protokol konektivitas yang diberikan.

Nilai yang didukung adalah:

- `pki`
- `password`

`connectivity.auth.credentials`

Kredensial yang digunakan untuk autentikasi.

`connectivity.auth.credentials.user`

Nama pengguna yang digunakan untuk masuk ke perangkat Anda.

`connectivity.auth.credentials.privKeyPath`

Jalur lengkap ke kunci pribadi yang digunakan untuk masuk ke perangkat Anda.

Nilai ini hanya berlaku jika `connectivity.auth.method` diatur ke `pki`.

`devices.connectivity.auth.credentials.password`

Kata sandi yang digunakan untuk masuk ke perangkat Anda.

Nilai ini hanya berlaku jika `connectivity.auth.method` diatur ke `password`.

> **Note**
>
> Tentukan `privKeyPath` hanya jika `method` diatur ke `pki`.
> Tentukan `password` hanya jika `method` diatur ke `password`.

Jalankan rangkaian tes

Setelah Anda membuat rangkaian tes Anda, Anda ingin memastikan bahwa rangkaian tes itu berfungsi seperti yang diharapkan. Selesaikan langkah-langkah berikut untuk menjalankan rangkaian pengujian dengan kolam perangkat yang sudah ada untuk melakukannya.

1. Salin folder `MyTestSuite_1.0.0` Anda ke dalam *`<device-tester-extract-location>`*`/tests`.

2. Jalankan perintah berikut:

```
cd <device-tester-extract-location>/bin
./devicetester_[linux | mac | win_x86-64] run-suite --suite-id MyTestSuite
```

IDT menjalankan rangkaian tes Anda dan mengalirkan hasilnya ke konsol. Ketika tes telah selesai berjalan, Anda akan melihat informasi berikut:

```
time="2020-10-19T09:24:47-07:00" level=info msg=Using pool: pool
time="2020-10-19T09:24:47-07:00" level=info msg=Using test suite "MyTestSuite_1.0.0"
 for execution
time="2020-10-19T09:24:47-07:00" level=info msg=b'hello world\n'
 suiteId=MyTestSuite groupId=myTestGroup testCaseId=myTestCase deviceId=my-device
 executionId=9a52f362-1227-11eb-86c9-8c8590419f30
time="2020-10-19T09:24:47-07:00" level=info msg=All tests finished.
 executionId=9a52f362-1227-11eb-86c9-8c8590419f30
time="2020-10-19T09:24:48-07:00" level=info msg=

========== Test Summary ==========
Execution Time:         1s
Tests Completed:        1
Tests Passed:           1
Tests Failed:           0
Tests Skipped:          0
---------------------------------
Test Groups:
    myTestGroup:        PASSED
---------------------------------
Path to IoT Device Tester Report: /path/to/devicetester/
results/9a52f362-1227-11eb-86c9-8c8590419f30/awsiotdevicetester_report.xml
Path to Test Execution Logs: /path/to/devicetester/
results/9a52f362-1227-11eb-86c9-8c8590419f30/logs
Path to Aggregated JUnit Report: /path/to/devicetester/
results/9a52f362-1227-11eb-86c9-8c8590419f30/MyTestSuite_Report.xml
```

Pemecahan Masalah

Gunakan informasi berikut untuk membantu menyelesaikan masalah dengan menyelesaikan tutorial.

Uji kasus tidak berjalan dengan sukses

Jika tes tidak berjalan sukses, IDT akan mengalirkan log kesalahan ke konsol yang dapat membantu Anda memecahkan masalah uji coba. Sebelum Anda memeriksa log kesalahan, verifikasi hal berikut:

- SDK Klien IDT berada dalam folder yang benar seperti yang dijelaskan dalam langkah ini.
- Pastikan Anda memenuhi semua prasyarat untuk tutorial ini.

Tidak dapat menyambung ke perangkat yang sedang diuji

Verifikasi hal berikut:

- File `device.json` Anda berisi alamat IP, port, dan informasi autentikasi yang benar.
- Anda dapat terhubung ke perangkat Anda melalui SSH dari komputer host Anda.

Buat file konfigurasi rangkaian tes IDT

Bagian ini menjelaskan format di mana Anda membuat file konfigurasi JSON yang Anda sertakan ketika Anda menyusun rangkaian uji kustom.

File JSON yang dibutuhkan

`suite.json`

Berisi informasi tentang rangkaian pengujian. Lihat Konfigurasikan suite.json.

`group.json`

Berisi informasi tentang grup uji. Anda harus membuat file `group.json` untuk setiap grup uji di rangkaian tes Anda. Lihat Konfigurasikan group.json.

`test.json`

Berisi informasi tentang grup uji. Anda harus membuat file `test.json` untuk setiap grup uji di rangkaian tes Anda. Lihat Konfigurasikan test.json.

File JSON opsional

`state_machine.json`

Menentukan bagaimana tes akan dijalankan ketika IDT menjalankan rangkaian tes. Lihat Konfigurasikan state_machine.json.

`userdata_schema.json`

Menentukan skema untuk file `userdata.json` yang dapat disertakan oleh test runner dalam konfigurasi pengaturannya. File `userdata.json` digunakan untuk konfigurasi informasi tambahan apa pun yang diperlukan untuk menjalankan tes tetapi tidak terdapat dalam file `device.json`. Lihat Konfigurasikan userdata_schema.json.

File konfigurasi JSON ditempatkan pada *<custom-test-suite-folder>* Anda seperti yang ditunjukkan di sini.

```
<custom-test-suite-folder>
### suite
    ### suite.json
    ### state_machine.json
    ### userdata_schema.json
    ### <test-group-folder>
        ### group.json
        ### <test-case-folder>
            ### test.json
```

Konfigurasikan suite.json

File `suite.json` menetapkan variabel lingkungan dan menentukan apakah data pengguna diperlukan untuk menjalankan rangkaian tes. Gunakan templat berikut untuk mengonfigurasi file *<custom-test-suite-folder>*`/suite/suite.json` Anda:

```
{
    "id": "<suite-name>_<suite-version>",
    "title": "<suite-title>",
    "details": "<suite-details>",
    "userDataRequired": true | false,
    "environmentVariables": [
        {
            "key": "<name>",
```

```
            "value": "<value>",
        },
        ...
        {
            "key": "<name>",
            "value": "<value>",
        }
    ]
}
```

Semua kolom yang berisi nilai wajib diisi seperti yang dijelaskan di sini:

`id`

ID unik yang ditetapkan pengguna untuk rangkaian uji. Nilai dari `id` harus cocok dengan nama folder rangkaian uji tempat file `suite.json` berada. Nama rangkaian dan versi rangkaian juga harus memenuhi persyaratan berikut:

- *<suite-name>* tidak dapat berisi garis bawah.
- *<suite-version>* dilambangkan sebagai *x.x.x*, di mana x adalah angka.

ID ditampilkan dalam laporan uji yang dihasilkan IDT.

`title`

Nama yang ditetapkan pengguna untuk produk atau fitur yang diuji oleh rangkaian tes ini. Nama ditampilkan dalam IDT CLI untuk test runner.

`details`

Deskripsi singkat tentang tujuan dari rangkaian tes.

`userDataRequired`

Menentukan apakah test runner perlu menyertakan informasi kustom dalam file `userdata.json`. Jika Anda menetapkan nilai ini ke `true`, Anda juga harus menyertakan file `userdata_schema.json` dalam folder rangkaian uji Anda.

`environmentVariables`

Tidak wajib. Serangkaian variabel lingkungan yang akan ditetapkan untuk rangkaian tes ini.

`environmentVariables.key`

Nama variabel lingkungan.

`environmentVariables.value`

Nilai variabel lingkungan.

Konfigurasikan group.json

File `group.json` menentukan apakah grup uji itu wajib atau opsional. Gunakan templat berikut untuk mengonfigurasi file *<custom-test-suite-folder>*`/suite/`*<test-group>*`/group.json` Anda:

```
{
    "id": "<group-id>",
    "title": "<group-title>",
    "details": "<group-details>",
    "optional": true | false,
}
```

Semua kolom yang berisi nilai wajib diisi seperti yang dijelaskan di sini:

`id`

ID unik yang ditetapkan pengguna untuk grup uji. Nilai dari`id`harus cocok dengan nama folder grup uji tempat`group.json`file terletak, dan tidak boleh berisi garis bawah (_). ID digunakan dalam laporan uji yang dihasilkan IDT.

`title`

Nama deskriptif untuk grup uji. Nama tersebut ditampilkan dalam IDT CLI untuk test runner.

`details`

Deskripsi singkat tentang tujuan dari grup tes.

`optional`

Tidak wajib. Atur ke `true` untuk menampilkan grup tes ini sebagai grup opsional setelah IDT selesai menjalankan tes yang diperlukan. Nilai defaultnya adalah `false`.

Konfigurasikan test.json

File `test.json` menentukan executable uji kasus dan variabel lingkungan yang digunakan oleh uji kasus. Untuk informasi selengkapnya tentang cara membuat executable uji kasus, lihat Buat executable uji kasus IDT.

Gunakan templat berikut untuk mengonfigurasi file *<custom-test-suite-folder>*/suite/*<test-group>*/*<test-case>*/test.json Anda:

```
{
    "id": "<test-id>",
    "title": "<test-title>",
    "details": "<test-details>",
    "requireDUT": true | false,
    "requiredResources": [
        {
            "name": "<resource-name>",
            "features": [
                {
                    "name": "<feature-name>",
                    "version": "<feature-version>",
                    "jobSlots": <job-slots>
                }
            ]
        }
    ],
    "execution": {
        "timeout": <timeout>,
        "mac": {
            "cmd": "/path/to/executable",
            "args": [
                "<argument>"
            ],
        },
        "linux": {
            "cmd": "/path/to/executable",
            "args": [
                "<argument>"
            ],
        },
        "win": {
            "cmd": "/path/to/executable",
            "args": [
```

```
                "<argument>"
            ]
        }
    },
    "environmentVariables": [
        {
            "key": "<name>",
            "value": "<value>",
        }
    ]
}
```

Semua kolom yang berisi nilai wajib diisi seperti yang dijelaskan di sini:

`id`

ID unik yang ditetapkan pengguna untuk grup uji. Nilai dari`id`harus cocok dengan nama folder uji kasus tempat`test.json`file terletak, dan tidak boleh berisi garis bawah (_). ID digunakan dalam laporan uji yang dihasilkan IDT.

`title`

Nama deskriptif untuk uji kasus. Nama tersebut ditampilkan dalam IDT CLI untuk test runner.

`details`

Deskripsi singkat tentang tujuan dari uji kasus.

`requireDUT`

Tidak wajib. Atur ke `true` jika perangkat diperlukan untuk menjalankan tes ini, jika tidak atur ke `false`. Nilai defaultnya adalah `true`. Test runner akan mengonfigurasi perangkat yang akan digunakannya untuk menjalankan pengujian di file `device.json`.

`requiredResources`

Tidak wajib. Serangkaian hal yang menyediakan informasi tentang perangkat sumber daya yang diperlukan untuk menjalankan tes ini.

`requiredResources.name`

Nama unik yang akan diberikan kepada sumber daya perangkat ketika tes ini berjalan.

`requiredResources.features`

Serangkaian fitur perangkat sumber daya yang ditetapkan pengguna.

`requiredResources.features.name`

Nama fitur. Fitur perangkat yang ingin Anda gunakan untuk perangkat ini. Nama ini dicocokkan dengan nama fitur yang disediakan oleh test runner di file `resource.json`.

`requiredResources.features.version`

Tidak wajib. Versi fitur. Nama ini dicocokkan dengan versi fitur yang disediakan oleh test runner di file `resource.json`. Jika versi tidak tersedia, maka fitur tersebut tidak dicentang. Jika nomor versi tidak diperlukan untuk fitur tersebut, biarkan kolom ini kosong.

`requiredResources.features.jobSlots`

Tidak wajib. Jumlah tes simultan yang dapat didukung fitur ini. Nilai default-nya adalah 1. Jika Anda ingin IDT menggunakan perangkat yang berbeda untuk masing-masing fitur, kami sarankan Anda menetapkan nilai ini ke 1.

`execution.timeout`

Jumlah waktu (dalam milidetik) yang ditunggu oleh IDT hingga tes tersebut selesai dijalankan. Untuk informasi selengkapnya tentang pengaturan parameter ini, lihat Buat executable uji kasus IDT.

`execution.os`

Executable uji kasus yang akan dijalankan berdasarkan sistem operasi komputer host yang menjalankan IDT. Nilai yang didukung adalah `linux`, `mac`, dan `win`.

`execution.os.cmd`

Jalur ke executable uji kasus yang ingin Anda jalankan untuk sistem operasi tertentu. Lokasi ini harus berada di jalur sistem.

`execution.os.args`

Tidak wajib. Argumen yang akan disediakan untuk menjalankan executable uji kasus.

`environmentVariables`

Tidak wajib. Serangkaian variabel lingkungan yang akan ditetapkan untuk uji kasus ini.

`environmentVariables.key`

Nama variabel lingkungan.

`environmentVariables.value`

Nilai variabel lingkungan.

> **Note**
>
> Jika Anda menentukan variabel lingkungan yang sama di file `test.json` dan di file `suite.json`, nilai dalam file `test.json` akan diutamakan.

Konfigurasikan state_machine.json

State machine adalah suatu konstruksi yang mengendalikan aliran eksekusi rangkaian uji. Ia menentukan keadaan awal dari rangkaian tes, mengelola transisi keadaan berdasarkan aturan yang ditetapkan pengguna, dan terus melakukan transisi melalui keadaan-keadaan tersebut sampai mencapai keadaan akhir.

Jika rangkaian tes Anda tidak menyertakan state machine yang ditetapkan pengguna, IDT akan membuat state machine untuk Anda. State machine default melakukan fungsi-fungsi berikut:

- Menyediakan test runner dengan kemampuan untuk memilih dan menjalankan grup uji tertentu, dan bukan seluruh rangkaian uji.
- Jika grup uji tertentu tidak dipilih, ia menjalankan setiap grup uji di rangkaian uji dengan urutan acak.
- Membuat laporan dan mencetak ringkasan konsol yang menunjukkan hasil tes untuk setiap grup uji dan uji kasus.

Untuk informasi selengkapnya tentang bagaimana state machine IDT bekerja, lihat Konfigurasikan mesin status IDT.

Konfigurasikan userdata_schema.json

File `userdata_schema.json` menentukan skema di mana test runner menyediakan data pengguna. Data pengguna diperlukan jika rangkaian uji Anda memerlukan informasi yang tidak ada di file `device.json`. Misalnya, pengujian Anda mungkin memerlukan kredensial jaringan Wi-Fi, port terbuka tertentu, atau sertifikat yang harus diberikan pengguna. Informasi ini dapat diberikan kepada IDT sebagai parameter input yang disebut `userdata`, nilai yang merupakan file `userdata.json`, yang dibuat oleh para pengguna dalam *`<device-tester-extract-location>`*`/config` mereka.

Format file `userdata.json` didasarkan pada `userdata_schema.json` yang Anda sertakan dalam rangkaian tes.

Untuk menunjukkan hal tersebut test runner harus menyediakan file `userdata.json`:

1. Di file `suite.json`, atur `userDataRequired` ke `true`.
2. Di *<custom-test-suite-folder>* Anda, buat file `userdata_schema.json`.
3. Edit file `userdata_schema.json` untuk membuat Skema IETF Draft v4 JSON yang valid.

Ketika IDT menjalankan rangkaian uji Anda, secara otomatis ia membaca skema dan menggunakannya untuk memvalidasi file `userdata.json` yang disediakan oleh test runner. Jika valid, isi file `userdata.json` akan tersedia baik di konteks IDT maupun di konteks state machine.

Konfigurasikan mesin status IDT

Mesin status adalah konstruksi yang mengontrol aliran eksekusi rangkaian pengujian. Ia menentukan keadaan awal dari rangkaian tes, mengelola transisi keadaan berdasarkan aturan yang ditetapkan pengguna, dan terus melakukan transisi melalui keadaan-keadaan tersebut sampai mencapai keadaan akhir.

Jika rangkaian tes Anda tidak menyertakan state machine yang ditetapkan pengguna, IDT akan membuat state machine untuk Anda. State machine default melakukan fungsi-fungsi berikut:

- Menyediakan test runner dengan kemampuan untuk memilih dan menjalankan grup uji tertentu, dan bukan seluruh rangkaian uji.
- Jika grup uji tertentu tidak dipilih, ia menjalankan setiap grup uji di rangkaian uji dengan urutan acak.
- Membuat laporan dan mencetak ringkasan konsol yang menunjukkan hasil tes untuk setiap grup uji dan uji kasus.

State machine untuk rangkaian uji IDT harus memenuhi kriteria berikut:

- Setiap keadaan sesuai dengan tindakan yang harus ddilakukan oleh IDT, seperti menjalankan grup uji atau produk file laporan.
- Transisi ke suatu keadaan akan menghasilkan tindakan yang terkait dengan keadaan tersebut.
- Setiap keadaan menentukan aturan transisi untuk keadaan berikutnya.
- Keadaan akhir harus berupa `Succeed` atau `Fail`.

Format state machine

Anda dapat menggunakan templat berikut untuk mengonfigurasi file *<custom-test-suite-folder>*/suite/state_machine.json Anda:

```
{
  "Comment": "<description>",
  "StartAt": "<state-name>",
  "States": {
    "<state-name>": {
      "Type": "<state-type>",
      // Additional state configuration
    }

    // Required states
    "Succeed": {
      "Type": "Succeed"
    },
    "Fail": {
      "Type": "Fail"
    }
  }
}
```

Semua kolom yang berisi nilai wajib diisi seperti yang dijelaskan di sini:

`Comment`

Sebuah deskripsi tentang state machine.

`StartAt`

Nama keadaan di mana IDT mulai menjalankan rangkaian tes. Nilai dari `StartAt` harus diatur ke salah satu keadaan yang tercantum di objek `States`.

`States`

Sebuah objek yang memetakan nama keadaan yang ditetapkan pengguna ke keadaan IDT yang valid. Setiap keadaan. Objek *nama-keadaan* berisi definisi keadaan valid yang dipetakan ke *nama-keadaan* tersebut.

Objek `States` harus mencakup keadaan `Succeed` dan `Fail`. Untuk informasi lebih lanjut tentang keadaan yang valid, lihat Keadaan yang valid dan definisi keadaan.

Keadaan yang valid dan definisi keadaan

Bagian ini menjelaskan definisi keadaan pada semua keadaan yang valid yang dapat digunakan dalam state machine IDT. Beberapa keadaan berikut mendukung konfigurasi pada tingkat uji kasus. Namun, kami sarankan Anda untuk mengonfigurasi aturan transisi keadaan pada tingkat grup uji dan bukan pada tingkat uji kasus kecuali jika benar-benar diperlukan.

Definisi keadaan

- RunTask
- Pilihan
- Paralel
- AddProductFeatures
- Laporan
- LogMessage
- SelectGroup
- Gagal
- Berhasil

RunTask

Keadaan `RunTask` menjalankan uji kasus dari grup uji yang ditentukan dalam rangkaian tes.

```
{
    "Type": "RunTask",
    "Next": "<state-name>",
    "TestGroup": "<group-id>",
    "TestCases": [
        "<test-id>"
    ],
    "ResultVar": "<result-name>"
}
```

Semua kolom yang berisi nilai wajib diisi seperti yang dijelaskan di sini:

`Next`

Nama keadaan yang akan ditransisi setelah melaksanakan tindakan dalam keadaan saat ini.

`TestGroup`

Tidak wajib. ID grup uji yang akan dijalankan. Jika nilai ini tidak ditentukan, IDT akan menjalankan grup uji yang dipilih oleh test runner.

`TestCases`

Tidak wajib. Serangkaian ID uji kasus dari grup yang ditentukan di `TestGroup`. Berdasarkan nilai-nilai `TestGroup` dan `TestCases`, IDT menentukan perilaku eksekusi tes sebagai berikut:

- Ketika `TestGroup` dan `TestCases` ditentukan, IDT akan menjalankan uji kasus tertentu dari grup uji.
- Ketika `TestCases` ditentukan tetapi `TestGroup` tidak ditentukan, IDT akan menjalankan uji kasus yang ditentukan.
- Ketika `TestGroup` ditentukan tetapi `TestCases` tidak ditentukan, IDT akan menjalankan semua uji kasus di grup uji yang ditentukan.
- Ketika `TestGroup` ataupun `TestCases` tidak ditentukan, IDT akan menjalankan semua uji kasus dari grup uji yang dipilih oleh test runner dari IDT CLI. Untuk mengaktifkan pilihan grup untuk test runner, Anda harus menyertakan keadaan `RunTask` dan `Choice` dalam file `statemachine.json` Anda. Untuk contoh cara kerjanya, lihatContoh mesin status: Jalankan grup uji yang dipilih pengguna.

  Untuk informasi selengkapnya tentang mengaktifkan perintah IDT CLI untuk test runner, lihat the section called “Aktifkan perintah IDT CLI”.

`ResultVar`

Nama variabel konteks yang akan diatur dengan hasil uji yang dijalankan. Jangan tentukan nilai ini jika Anda tidak menentukan nilai untuk `TestGroup`. IDT menetapkan nilai variabel yang Anda tentukan di `ResultVar` hingga `true` atau `false` berdasarkan berikut ini:

- Jika nama variabel adalah dari bentuk *`text`*`_`*`text`*`_passed`, maka nilainya akan diatur ke apakah semua tes dalam grup uji pertama akan dilalui atau dilompati.
- Dalam semua kasus lainnya, nilai akan diatur ke apakah semua tes di semua grup uji akan dilalui atau dilompati.

Biasanya, Anda akan menggunakan keadaan `RunTask` untuk menentukan ID grup uji tanpa menentukan ID uji kasus individu, sehingga IDT akan menjalankan semua uji kasus dalam grup uji tertentu. Semua uji kasus yang dijalankan oleh keadaan ini berjalan secara paralel, dengan

urutan acak. Namun, jika semua uji kasus memerlukan perangkat untuk dijalankan, dan hanya satu perangkat yang tersedia, maka uji kasus akan berjalan secara berurutan sebagai gantinya.

Penanganan kesalahan

Jika salah satu grup uji atau ID uji kasus tertentu tidak valid, maka keadaan ini akan mengeluarkan kesalahan eksekusi `RunTaskError`. Jika keadaan ini menemukan kesalahan eksekusi, ia juga akan menetapkan variabel `hasExecutionError` dalam konteks state machine ke `true`.

Pilihan

Keadaan `Choice` memungkinkan Anda secara dinamis mengatur keadaan berikutnya yang akan ditransisikan berdasarkan keadaan yang ditetapkan pengguna.

```
{
    "Type": "Choice",
    "Default": "<state-name>",
    "FallthroughOnError": true | false,
    "Choices": [
        {
            "Expression": "<expression>",
            "Next": "<state-name>"
        }
    ]
}
```

Semua kolom yang berisi nilai wajib diisi seperti yang dijelaskan di sini:

`Default`

Keadaan default yang akan ditransisikan jika tidak terdapat ekspresi yang ditentukan di `Choices` dapat dievaluasi pada `true`.

`FallthroughOnError`

Tidak wajib. Menentukan perilaku ketika keadaan tersebut bertemu kesalahan dalam mengevaluasi ekspresi. Atur ke `true` jika Anda ingin melompati ekspresi jika hasil evaluasi menghasilkan kesalahan. Jika tidak ada ekspresi yang cocok, state machine akan bertransisi ke keadaan `Default`. Jika nilai `FallthroughOnError` tidak ditentukan, default-nya adalah `false`.

`Choices`

Serangkaian ekspresi dan keadaan untuk menentukan keadaan mana yang akan ditransisikan setelah mengeksekusi tindakan dalam keadaan saat ini.

`Choices.Expression`

Ekspresi yang harus dievaluasi pada nilai boolean. Jika ekspresi mengevaluasi `true`, maka state machine akan bertransisi ke keadaan yang ditentukan dalam `Choices.Next`. String ekspresi mengambil nilai-nilai dari konteks state machine dan kemudian melakukan operasi padanya untuk sampai pada nilai boolean. Untuk informasi tentang mengakses konteks state machine, lihat Konteks mesin keadaan.

`Choices.Next`

Nama keadaan yang akan ditransisikan jika ekspresi yang ditentukan dalam `Choices.Expression` dievaluasi pada `true`.

Penanganan kesalahan

Keadaan `Choice` dapat memerlukan penanganan kesalahan dalam kasus berikut:

- Beberapa variabel dalam ekspresi pilihan tidak ada dalam konteks state machine.
- Hasil ekspresi bukan merupakan nilai boolean.
- Hasil pencarian JSON bukanlah string, nomor, atau boolean.

Anda tidak dapat menggunakan blok `Catch` untuk menangani kesalahan dalam keadaan ini. Jika Anda ingin berhenti mengeksekusi state machine ketika menemukan kesalahan, Anda harus mengatur `FallthroughOnError` ke `false`. Namun, kami menyarankan agar Anda mengatur `FallthroughOnError` ke `true` Anda, dan tergantung pada kasus penggunaan Anda, lakukan salah satu langkah berikut:

- Jika variabel yang Anda akses diharapkan untuk tidak ada dalam beberapa kasus, gunakan nilai `Default` dan blok `Choices` tambahan untuk menentukan keadaan berikutnya.
- Jika variabel yang Anda akses harus selalu ada, atur keadaan `Default`ke `Fail`.

Paralel

Keadaan `Parallel` memungkinkan Anda untuk menentukan dan menjalankan state machine baru secara paralel satu sama lain.

```
{
    "Type": "Parallel",
```

```
    "Next": "<state-name>",
    "Branches": [
        <state-machine-definition>
    ]
}
```

Semua kolom yang berisi nilai wajib diisi seperti yang dijelaskan di sini:

`Next`

Nama keadaan yang akan ditransisi setelah melaksanakan tindakan dalam keadaan saat ini.

`Branches`

Serangkaian definisi state machine yang akan dijalankan. Setiap definisi state machine harus berisi keadaan `StartAt`, `Succeed`, dan `Fail` miliknya sendiri. Definisi state machine dalam rangkaian ini tidak dapat mengacu pada keadaan di luar definisinya sendiri.

> **Note**
>
> Karena setiap cabang state machine memiliki konteks state machine yang sama, pengaturan variabel dalam satu cabang dan kemudian pembacaan variabel-variabel dari cabang lain dapat mengakibatkan perilaku yang tidak terduga.

Keadaan `Parallel` bergerak ke keadaan berikutnya hanya setelah keadaan tersebut menjalankan semua cabang state machine. Setiap keadaan yang memerlukan perangkat akan menunggu untuk berjalan hingga perangkat tersebut tersedia. Jika beberapa perangkat tersedia, keadaan ini akan menjalankan uji kasus dari beberapa grup secara paralel. Jika tidak tersedia perangkat yang memadai, uji kasus akan berjalan secara berurutan. Karena uji kasus dijalankan dalam urutan acak ketika berjalan secara paralel, perangkat yang berbeda mungkin digunakan untuk menjalankan tes dari grup tes yang sama.

Penanganan kesalahan

Pastikan bahwa baik state machine cabang dan state machine induk bertransisi ke keadaan `Fail` untuk menangani kesalahan eksekusi.

Karena state machine cabang tidak mengirimkan kesalahan eksekusi ke state machine induk, Anda tidak dapat menggunakan blok `Catch` untuk menangani kesalahan eksekusi di state machine cabang. Sebagai gantinya, gunakan nilai `hasExecutionErrors` dalam konteks state machine

bersama. Untuk contoh cara bekerjanya, lihat Contoh mesin status: Jalankan dua grup uji secara parallel.

AddProductFeatures

Keadaan `AddProductFeatures` memungkinkan Anda menambahkan fitur produk ke file `awsiotdevicetester_report.xml` yang dihasilkan oleh IDT.

Fitur produk adalah informasi yang ditetapkan pengguna tentang kriteria spesifik yang mungkin dipenuhi oleh perangkat. Misalnya, fitur produk `MQTT` dapat menetapkan bahwa perangkat akan menerbitkan pesan MQTT dengan benar. Dalam laporan tersebut, fitur produk ditetapkan sebagai `supported`, `not-supported`, atau nilai kustom, berdasarkan apakah tes yang ditentukan berhasil dilalui.

> **Note**
>
> Keadaan `AddProductFeatures` tidak menghasilkan laporan dengan sendirinya. Keadaan ini harus bertransisi ke keadaan `Report` untuk menghasilkan laporan.

```
{
    "Type": "Parallel",
    "Next": "<state-name>",
    "Features": [
        {
            "Feature": "<feature-name>",
            "Groups": [
                "<group-id>"
            ],
            "OneOfGroups": [
                "<group-id>"
            ],
            "TestCases": [
                "<test-id>"
            ],
            "IsRequired": true | false,
            "ExecutionMethods": [
                "<execution-method>"
            ]
        }
```

```
    ]
}
```

Semua kolom yang berisi nilai wajib diisi seperti yang dijelaskan di sini:

`Next`

Nama keadaan yang akan ditransisi setelah melaksanakan tindakan dalam keadaan saat ini.

`Features`

Serangkaian fitur produk yang akan ditampilkan di file `awsiotdevicetester_report.xml`.

`Feature`

Nama fitur

`FeatureValue`

Tidak wajib. Nilai kustom yang akan digunakan dalam laporan, dan bukan `supported`. Jika nilai ini tidak ditentukan, maka berdasarkan hasil tes, nilai fitur akan diatur ke `supported` atau `not-supported`.

Jika Anda menggunakan nilai kustom untuk `FeatureValue`, Anda dapat menguji fitur yang sama dengan kondisi yang berbeda, dan IDT akan menggabungkan nilai fitur untuk kondisi yang didukung. Misalnya, petikan berikut menunjukkan fitur `MyFeature` dengan dua nilai fitur yang terpisah:

```
...
{
    "Feature": "MyFeature",
    "FeatureValue": "first-feature-supported",
    "Groups": ["first-feature-group"]
},
{
    "Feature": "MyFeature",
    "FeatureValue": "second-feature-supported",
    "Groups": ["second-feature-group"]
},
...
```

Jika kedua grup uji itu lulus, nilai fitur akan diatur ke `first-feature-supported, second-feature-supported`.

`Groups`

Tidak wajib. Serangkaian ID grup uji. Semua tes dalam setiap kelompok uji yang ditentukan harus lulus pada fitur yang akan didukung.

`OneOfGroups`

Tidak wajib. Serangkaian ID grup uji. Semua tes dalam setidaknya satu kelompok uji yang ditentukan harus lulus pada fitur yang akan didukung.

`TestCases`

Tidak wajib. Serangkaian ID grup uji. Jika Anda menentukan nilai ini, maka hal berikut ini akan berlaku:

- Semua uji kasus yang ditentukan harus lulus pada fitur yang akan didukung.
- `Groups` harus berisi hanya satu ID grup uji.
- `OneOfGroups` tidak boleh ditentukan.

`IsRequired`

Tidak wajib. Atur ke `false` untuk menandai fitur ini sebagai fitur opsional dalam laporan. Nilai default adalah `true`.

`ExecutionMethods`

Tidak wajib. Serangkaian metode eksekusi yang sesuai dengan nilai `protocol` yang ditentukan dalam file `device.json`. Jika nilai ini ditentukan, test runner harus menentukan nilai `protocol` yang cocok dengan salah satu nilai dalam rangkaian ini untuk menyertakan fitur tersebut dalam laporan. Jika nilai ini tidak ditentukan, maka fitur itu akan selalu disertakan dalam laporan.

Untuk menggunakan keadaan `AddProductFeatures`, Anda harus menetapkan nilai `ResultVar` di keadaan `RunTask` ke salah satu nilai berikut:

- Jika Anda telah menentukan ID uji kasus individu, atur `ResultVar` ke *`group-id_test-id`*`_passed`.
- Jika Anda tidak menentukan ID uji kasus individu, atur `ResultVar` ke *`group-id`*`_passed`.

Keadaan `AddProductFeatures` akan mengecek hasil tes dengan cara berikut:

- Jika Anda tidak menentukan ID uji kasus, maka hasil untuk setiap grup uji akan ditentukan dari nilai variabel *group-id*_passed dalam konteks state machine.
- Jika Anda tidak menentukan ID uji kasus, maka hasil untuk setiap tes akan ditentukan dari nilai variabel *group-id_test-id*_passed dalam konteks state machine.

Penanganan kesalahan

Jika ID grup yang disediakan dalam keadaan ini bukan ID grup yang valid, maka keadaan ini akan menghasilkan kesalahan eksekusi `AddProductFeaturesError`. Jika keadaan ini menemukan kesalahan eksekusi, ia juga akan menetapkan variabel `hasExecutionErrors` dalam konteks state machine ke `true`.

Laporan

Keadaan `Report` menghasilkan file *suite-name*_`Report.xml` dan `awsiotdevicetester_report.xml`. Keadaan ini juga mengalirkan laporan ke konsol.

```
{
    "Type": "Report",
    "Next": "<state-name>"
}
```

Semua kolom yang berisi nilai wajib diisi seperti yang dijelaskan di sini:

`Next`

Nama keadaan yang akan ditransisi setelah melaksanakan tindakan dalam keadaan saat ini.

Anda harus selalu beralih ke keadaan `Report` menjelang akhir aliran eksekusi tes agar test runner dapat melihat hasil tes. Biasanya, keadaan berikutnya setelah keadaan ini adalah `Succeed`.

Penanganan kesalahan

Jika keadaan ini mengalami masalah dalam menghasilkan laporan, keadaan tersebut akan mengeluarkan kesalahan eksekusi `ReportError`.

LogMessage

Keadaan `LogMessage` akan menghasilkan file `test_manager.log` dan mengalirkan pesan log ke konsol.

```
{
    "Type": "LogMessage",
    "Next": "<state-name>"
    "Level": "info | warn | error"
    "Message": "<message>"
}
```

Semua kolom yang berisi nilai wajib diisi seperti yang dijelaskan di sini:

`Next`

Nama keadaan yang akan ditransisi setelah melaksanakan tindakan dalam keadaan saat ini.

`Level`

Tingkat kesalahan tempat membuat pesan log. Jika Anda menentukan tingkat yang tidak valid, keadaan ini akan menghasilkan pesan kesalahan dan membuangnya.

`Message`

Pesan yang akan dicatat.

SelectGroup

Keadaan `SelectGroup` memperbarui konteks state machine untuk menunjukkan grup mana yang dipilih. Nilai-nilai yang ditetapkan oleh keadaan ini digunakan oleh setiap kondisi `Choice` berikutnya.

```
{
    "Type": "SelectGroup",
    "Next": "<state-name>"
    "TestGroups": [
        <group-id>"
    ]
}
```

Semua kolom yang berisi nilai wajib diisi seperti yang dijelaskan di sini:

`Next`

Nama keadaan yang akan ditransisi setelah melaksanakan tindakan dalam keadaan saat ini.

`TestGroups`

Serangkaian grup uji yang akan ditandai sudah dipilih. Untuk setiap ID grup uji dalam rangkaian ini, variabel *`group-id`*`_selected` akan diatur ke `true` dalam konteks. Pastikan bahwa Anda memberikan ID grup tes yang valid karena IDT tidak memvalidasi apakah grup tertentu ada.

Gagal

Keadaan `Fail` menunjukkan bahwa state machine tidak mengeksekusi dengan benar. Ini adalah keadaan akhir untuk state machine, dan setiap definisi state machine harus mencakup keadaan ini.

```
{
    "Type": "Fail"
}
```

Berhasil

Keadaan `Succeed` menunjukkan bahwa state machine mengeksekusi dengan benar. Ini adalah keadaan akhir untuk state machine, dan setiap definisi state machine harus mencakup keadaan ini.

```
{
    "Type": "Succeed"
}
```

Konteks mesin keadaan

Konteks state machine adalah dokumen JSON baca-saja yang berisi data yang tersedia untuk state machine selama eksekusi. Konteks state machine hanya dapat diakses dari state machine, dan berisi informasi yang menentukan aliran uji. Misalnya, Anda dapat menggunakan informasi yang dikonfigurasi oleh test runner di file `userdata.json` untuk menentukan apakah pengujian tertentu wajib dijalankan.

Konteks state machine menggunakan format berikut:

```
{
    "pool": {
        <device-json-pool-element>
    },
    "userData": {
        <userdata-json-content>
    },
```

```
    "config": {
        <config-json-content>
    },
    "suiteFailed": true | false,
    "specificTestGroups": [
        "<group-id>"
    ],
    "specificTestCases": [
        "<test-id>"
    ],
    "hasExecutionErrors": true
}
```

`pool`

Informasi tentang kolam perangkat yang dipilih untuk uji coba. Untuk kolam perangkat yang dipilih, informasi ini diambil dari elemen rangkaian perangkat tingkat atas yang sesuai yang ditentukan dalam file `device.json`.

`userData`

Informasi di file `userdata.json`.

`config`

Informasi menyematkan file `config.json`.

`suiteFailed`

Nilai diatur ke `false` ketika state machine dimulai. Jika grup uji gagal dalam keadaan `RunTask`, maka nilai ini akan ditetapkan ke `true` untuk durasi sisa eksekusi state machine.

`specificTestGroups`

Jika test runner memilih grup uji tertentu yang akan dijalankan dan bukan keseluruhan rangkaian uji, kunci ini akan ini dibuat dan berisi daftar ID grup uji tertentu.

`specificTestCases`

Jika test runner memilih grup uji tertentu yang akan dijalankan dan bukan keseluruhan rangkaian uji, kunci ini akan dibuat dan berisi daftar ID uji kasus tertentu.

`hasExecutionErrors`

Tidak keluar saat state machine dimulai. Jika keadaan apa pun menemukan kesalahan eksekusi, variabel ini akan dibuat dan diatur ke `true` selama durasi sisa eksekusi state machine.

Anda dapat melakukan kueri atas konteks tersebut dengan menggunakan notasi JSONPath. Sintaks untuk kueri JSONPath dalam definisi keadaan adalah `{{$.query}}`. Anda dapat menggunakan kueri JSONPath sebagai string placeholder dalam beberapa keadaan. IDT menggantikan string placeholder dengan nilai kueri JSONPath yang dievaluasi dari konteks. Anda dapat menggunakan placeholder untuk nilai-nilai berikut:

- Nilai `TestCases` dalam keadaan `RunTask`.
- Nilai `Expression` keadaan `Choice`.

Ketika Anda mengakses data dari konteks state machine, pastikan keadaan berikut dipenuhi:

- Jalur JSON Anda harus dimulai dengan `$.`
- Setiap nilai harus dievaluasi pada string, angka, atau boolean.

Untuk informasi lebih lanjut tentang peggunaan notasi JSONPath untuk mengakses data dari konteks, lihat Gunakan konteks IDT.

Kesalahan eksekusi

Kesalahan eksekusi adalah kesalahan dalam definisi state machine yang ditemui oleh state machine mesin ketika mengeksekusi keadaan. IDT mencatat informasi tentang setiap kesalahan dalam file `test_manager.log` dan mengalirkan pesan log ke konsol.

Anda dapat menggunakan metode berikut untuk menangani kesalahan eksekusi:

- Tambahkan blok `Catch` dalam definisi keadaan.
- Periksa nilai dari nilai `hasExecutionErrors` dalam konteks state machine.

Tangkap

Untuk menggunakan `Catch`, tambahkan hal berikut ini ke definisi keadaan Anda:

```
"Catch": [
    {
        "ErrorEquals": [
            "<error-type>"
        ]
        "Next": "<state-name>"
```

```
    }
]
```

Semua kolom yang berisi nilai wajib diisi seperti yang dijelaskan di sini:

`Catch.ErrorEquals`

Serangkaian jenis kesalahan yang akan ditangkap. Jika kesalahan eksekusi cocok dengan salah satu nilai yang ditentukan, maka state machine akan bertransisi ke keadaan yang ditentukan dalam `Catch.Next`. Lihat setiap definisi keadaan untuk informasi tentang jenis kesalahan yang dihasilkannya.

`Catch.Next`

Keadaan berikutnya yang akan ditransisikan jika keadaan saat ini menemukan kesalahan eksekusi yang cocok dengan salah satu nilai yang ditentukan dalam `Catch.ErrorEquals`.

Blok tangkapan ditangani secara berurutan hingga salah satunya cocok. Jika tidak ada kesalahan yang cocok dengan yang tercantum dalam blok Tangkapan, maka state machine akan terus mengeksekusi. Karena kesalahan eksekusi adalah akibat dari definisi keadaan yang salah, kami sarankan Anda beralih ke keadaan gagal ketika suatu keadaan mengalami kesalahan eksekusi.

HasExecutionError

Ketika beberapa keadaan mengalami kesalahan eksekusi, selain mengeluarkan kesalahan, keaadaan itu juga mengatur nilai `hasExecutionError` ke `true` dalam konteks state machine. Anda dapat menggunakan nilai ini untuk mendeteksi ketika terjadi kesalahan, dan kemudian menggunakan keadaan `Choice` untuk mengalihkan state machine ke keadaan `Fail`.

Metode ini memiliki karakteristik sebagai berikut.

- State machine tidak dimulai dengan nilai yang ditugaskan pada `hasExecutionError`, dan nilai ini tidak tersedia sampai keadaan tertentu menetapkannya. Ini berarti bahwa Anda harus secara tegas mengatur `FallthroughOnError` ke `false` untuk keadaan `Choice` yang mengakses nilai ini untuk mencegah state machine berhenti jika tidak ada kesalahan eksekusi yang terjadi.
- Setelah ditetapkan ke `true`, `hasExecutionError` tidak pernah ditetapkan menjadi salah atau dihapus dari konteks. Ini berarti bahwa nilai ini berguna hanya pertama kalinya ketika nilai tersebut ditetapkan ke `true`, dan untuk semua keadaan berikutnya, nilai itu tidak memberikan nilai yang berarti.

- Nilai `hasExecutionError` dibagi dengan semua cabang state machine pada keadaan `Parallel`, yang dapat mengakibatkan hasil yang tidak diharapkan tergantung pada urutan yang diakses.

Karena karakteristik ini, kami tidak menyarankan Anda menggunakan metode ini jika Anda dapat menggunakan blok Catch sebagai gantinya.

Contoh mesin keadaan

Bagian ini menyediakan beberapa contoh konfigurasi state machine.

Contoh

- Contoh mesin status: Jalankan grup uji tunggal
- Contoh mesin status: Jalankan grup uji yang dipilih pengguna
- Contoh mesin status: Jalankan grup uji tunggal dengan fitur produk
- Contoh mesin status: Jalankan dua grup uji secara parallel

Contoh mesin status: Jalankan grup uji tunggal

State machine ini:

- Menjalankan grup uji dengan id `GroupA`, yang harus ada dalam rangkaian pada file `group.json`.
- Memeriksa kesalahan eksekusi dan bertransisi ke `Fail` jika ada yang ditemukan.
- Menghasilkan laporan dan bertransisi ke `Succeed` jika tidak ada kesalahan, dan `Fail` bila sebaliknya.

```
{
    "Comment": "Runs a single group and then generates a report.",
    "StartAt": "RunGroupA",
    "States": {
        "RunGroupA": {
            "Type": "RunTask",
            "Next": "Report",
            "TestGroup": "GroupA",
            "Catch": [
                {
                    "ErrorEquals": [
```

```
                        "RunTaskError"
                    ],
                    "Next": "Fail"
                }
            ]
        },
        "Report": {
            "Type": "Report",
            "Next": "Succeed",
            "Catch": [
                {
                    "ErrorEquals": [
                        "ReportError"
                    ],
                    "Next": "Fail"
                }
            ]
        },
        "Succeed": {
            "Type": "Succeed"
        },
        "Fail": {
            "Type": "Fail"
        }
    }
}
```

Contoh mesin status: Jalankan grup uji yang dipilih pengguna

State machine ini:

- Memeriksa apakah test runner telah memilih grup uji tertentu. State machine tidak memeriksa uji kasus tertentu karena test runner tidak dapat memilih uji kasus tanpa sekaligus memilih grup uji.
- Jika grup uji sudah dipilih:
  - Jalankan uji kasus dalam grup uji yang dipilih. Untuk melakukannya, state machine tidak secara tegas menentukan grup uji atau uji kasus di keadaan `RunTask`
  - Buat laporan setelah menjalankan semua tes dan keluar.
- Jika grup uji tidak dipilih:
  - Jalankan tes dalam grup uji `GroupA`.
  - Buat laporan dan keluar.

```
{
    "Comment": "Runs specific groups if the test runner chose to do that, otherwise
 runs GroupA.",
    "StartAt": "SpecificGroupsCheck",
    "States": {
        "SpecificGroupsCheck": {
            "Type": "Choice",
            "Default": "RunGroupA",
            "FallthroughOnError": true,
            "Choices": [
                {
                    "Expression": "{{$.specificTestGroups[0]}} != ''",
                    "Next": "RunSpecificGroups"
                }
            ]
        },
        "RunSpecificGroups": {
            "Type": "RunTask",
            "Next": "Report",
            "Catch": [
                {
                    "ErrorEquals": [
                        "RunTaskError"
                    ],
                    "Next": "Fail"
                }
            ]
        },
        "RunGroupA": {
            "Type": "RunTask",
            "Next": "Report",
            "TestGroup": "GroupA",
            "Catch": [
                {
                    "ErrorEquals": [
                        "RunTaskError"
                    ],
                    "Next": "Fail"
                }
            ]
        },
        "Report": {
            "Type": "Report",
```

```
            "Next": "Succeed",
            "Catch": [
                {
                    "ErrorEquals": [
                        "ReportError"
                    ],
                    "Next": "Fail"
                }
            ]
        },
        "Succeed": {
            "Type": "Succeed"
        },
        "Fail": {
            "Type": "Fail"
        }
    }
}
```

Contoh mesin status: Jalankan grup uji tunggal dengan fitur produk

State machine ini:

- Menjalankan grup uji `GroupA`.
- Memeriksa kesalahan eksekusi dan bertransisi ke `Fail` jika ada yang ditemukan.
- Menambahkan fitur `FeatureThatDependsOnGroupA` pada file `awsiotdevicetester_report.xml`:
  - Jika `GroupA` lulus, fitur tersebut diatur ke `supported`.
  - Fitur ini tidak ditandai opsional dalam laporan.
- Menghasilkan laporan dan bertransisi ke `Succeed` jika tidak ada kesalahan, dan `Fail` bila sebaliknya.

```
{
    "Comment": "Runs GroupA and adds product features based on GroupA",
    "StartAt": "RunGroupA",
    "States": {
        "RunGroupA": {
            "Type": "RunTask",
            "Next": "AddProductFeatures",
            "TestGroup": "GroupA",
```

```
            "ResultVar": "GroupA_passed",
            "Catch": [
                {
                    "ErrorEquals": [
                        "RunTaskError"
                    ],
                    "Next": "Fail"
                }
            ]
        },
        "AddProductFeatures": {
            "Type": "AddProductFeatures",
            "Next": "Report",
            "Features": [
                {
                    "Feature": "FeatureThatDependsOnGroupA",
                    "Groups": [
                        "GroupA"
                    ],
                    "IsRequired": true
                }
            ]
        },
        "Report": {
            "Type": "Report",
            "Next": "Succeed",
            "Catch": [
                {
                    "ErrorEquals": [
                        "ReportError"
                    ],
                    "Next": "Fail"
                }
            ]
        },
        "Succeed": {
            "Type": "Succeed"
        },
        "Fail": {
            "Type": "Fail"
        }
    }
}
```

Contoh mesin status: Jalankan dua grup uji secara parallel

State machine ini:

- Menjalankan grup tes `GroupA` dan `GroupB` secara paralel. Variabel `ResultVar` yang disimpan dalam konteks tersebut oleh keadaan `RunTask` dalam state machine cabang yang tersedia pada keadaan `AddProductFeatures`
- Memeriksa kesalahan eksekusi dan bertransisi ke `Fail` jika ada yang ditemukan. State machine ini tidak menggunakan blok `Catch` karena metode itu tidak mendeteksi kesalahan eksekusi di state machine cabang.
- Menambahkan fitur ke file `awsiotdevicetester_report.xml` berdasarkan grup-grup yang lulus
  - Jika `GroupA` lulus, fitur tersebut diatur ke `supported`.
  - Fitur ini tidak ditandai opsional dalam laporan.
- Menghasilkan laporan dan bertransisi ke `Succeed` jika tidak ada kesalahan, dan `Fail` bila sebaliknya.

Jika dua perangkat dikonfigurasi di kolam perangkat, baik `GroupA` maupun `GroupB` dapat berjalan pada saat yang sama. Namun, jika `GroupA` atau `GroupB` memiliki beberapa tes di dalamnya, maka kedua perangkat dapat dialokasikan pada tes tersebut. Jika hanya satu perangkat yang dikonfigurasi, grup uji akan berjalan secara berurutan.

```
{
    "Comment": "Runs GroupA and GroupB in parallel",
    "StartAt": "RunGroupAAndB",
    "States": {
        "RunGroupAAndB": {
            "Type": "Parallel",
            "Next": "CheckForErrors",
            "Branches": [
                {
                    "Comment": "Run GroupA state machine",
                    "StartAt": "RunGroupA",
                    "States": {
                        "RunGroupA": {
                            "Type": "RunTask",
                            "Next": "Succeed",
                            "TestGroup": "GroupA",
                            "ResultVar": "GroupA_passed",
```

```
                        "Catch": [
                            {
                                "ErrorEquals": [
                                    "RunTaskError"
                                ],
                                "Next": "Fail"
                            }
                        ]
                    },
                    "Succeed": {
                        "Type": "Succeed"
                    },
                    "Fail": {
                        "Type": "Fail"
                    }
                }
            },
            {
                "Comment": "Run GroupB state machine",
                "StartAt": "RunGroupB",
                "States": {
                    "RunGroupA": {
                        "Type": "RunTask",
                        "Next": "Succeed",
                        "TestGroup": "GroupB",
                        "ResultVar": "GroupB_passed",
                        "Catch": [
                            {
                                "ErrorEquals": [
                                    "RunTaskError"
                                ],
                                "Next": "Fail"
                            }
                        ]
                    },
                    "Succeed": {
                        "Type": "Succeed"
                    },
                    "Fail": {
                        "Type": "Fail"
                    }
                }
            }
        ]
```

```
        },
        "CheckForErrors": {
            "Type": "Choice",
            "Default": "AddProductFeatures",
            "FallthroughOnError": true,
            "Choices": [
                {
                    "Expression": "{{$.hasExecutionErrors}} == true",
                    "Next": "Fail"
                }
            ]
        },
        "AddProductFeatures": {
            "Type": "AddProductFeatures",
            "Next": "Report",
            "Features": [
                {
                    "Feature": "FeatureThatDependsOnGroupA",
                    "Groups": [
                        "GroupA"
                    ],
                    "IsRequired": true
                },
                {
                    "Feature": "FeatureThatDependsOnGroupB",
                    "Groups": [
                        "GroupB"
                    ],
                    "IsRequired": true
                }
            ]
        },
        "Report": {
            "Type": "Report",
            "Next": "Succeed",
            "Catch": [
                {
                    "ErrorEquals": [
                        "ReportError"
                    ],
                    "Next": "Fail"
                }
            ]
        },
```

```
        "Succeed": {
            "Type": "Succeed"
        },
        "Fail": {
            "Type": "Fail"
        }
    }
}
```

Buat executable uji kasus IDT

Anda dapat membuat dan menempatkan executable uji kasus dalam folder rangkaian tes dengan cara berikut:

- Untuk rangkaian tes yang menggunakan argumen atau variabel lingkungan dari file `test.json` untuk menentukan tes mana yang akan dijalankan, Anda dapat membuat uji kasus tunggal yang dapat dieksekusi untuk seluruh rangkaian tes, atau tes yang dapat dijalankan untuk setiap grup uji di rangkaian tes.
- Untuk rangkaian tes di mana Anda ingin menjalankan tes tertentu berdasarkan perintah tertentu, Anda membuat satu executable uji kasus untuk setiap uji kasus di rangkaian tes.

Sebagai penyusun tes, Anda dapat menentukan pendekatan yang sesuai untuk kasus penggunaan Anda dan menyusun executable uji kasus yang sesuai. Pastikan bahwa Anda menyediakan jalur eksekusi uji kasus yang benar di setiap file `test.json`, dan bahwa executable yang ditentukan berjalan dengan benar.

Ketika semua perangkat siap untuk dijalankan oleh uji kasus, IDT akan membaca file-file berikut:

- `test.json` untuk uji kasus yang dipilih menentukan proses yang akan dimulai dan variabel lingkungan yang akan diatur.
- `suite.json` untuk rangkaian uji tersebut menentukan variabel lingkungan yang akan diatur.

IDT memulai proses executable pengujian yang diperlukan berdasarkan perintah dan argumen yang ditentukan dalam file `test.json`, dan melewati variabel lingkungan yang diperlukan untuk proses tersebut.

Gunakan IDT Client SDK

IDT Client SDK memungkinkan Anda cara Anda menulis logika uji di executable tes Anda dengan perintah API yang dapat Anda gunakan untuk berinteraksi dengan IDT dan perangkat Anda yang sedang diuji. IDT saat ini menyediakan SDK berikut:

- IDT Client SDK for Python
- IDT Client SDK for Go

SDK ini terletak di folder *<device-tester-extract-location>*/sdks. Ketika Anda membuat executable uji kasus yang baru, Anda harus menyalin SDK yang ingin Anda gunakan ke folder yang berisi executable uji kasus dan mengacu pada SDK dalam kode Anda. Bagian ini memberikan penjelasan singkat tentang perintah API yang tersedia yang dapat Anda gunakan dalam executable uji kasus Anda.

Dalam Bagian Ini

- Interaksi perangkat
- Interaksi IDT
- Interaksi host

Interaksi perangkat

Perintah berikut memungkinkan Anda untuk berkomunikasi dengan perangkat yang diuji tanpa harus menerapkan interaksi perangkat tambahan dan fungsi manajemen konektivitas apa pun.

`ExecuteOnDevice`

Memungkinkan rangkaian tes untuk menjalankan perintah shell pada perangkat yang mendukung SSH atau koneksi Docker shell.

`CopyToDevice`

Memungkinkan rangkaian tes untuk menyalin file lokal dari mesin host yang menjalankan IDT ke lokasi yang ditentukan pada perangkat yang mendukung SSH atau koneksi Docker shell.

`ReadFromDevice`

Memungkinkan rangkaian tes untuk membaca dari port serial perangkat yang mendukung koneksi UART.

> **Note**
>
> Karena IDT tidak mengelola koneksi langsung ke perangkat yang dibuat menggunakan informasi akses perangkat dari konteks, sebaiknya gunakan perintah API interaksi perangkat ini di executable uji kasus. Namun, jika perintah ini tidak memenuhi persyaratan uji kasus Anda, maka Anda dapat mengambil informasi akses perangkat dari konteks IDT dan menggunakannya untuk membuat koneksi langsung ke perangkat dari rangkaian tes. Untuk membuat sambungan langsung, ambil informasi di `device.connectivity` dan `resource.devices.connectivity` masing-masing untuk perangkat Anda yang sedang diuji dan untuk perangkat sumber daya. Untuk informasi lebih lanjut mengenai penggunaan konteks IDT, lihat Gunakan konteks IDT.

Interaksi IDT

Perintah berikut memungkinkan rangkaian tes Anda untuk berkomunikasi dengan IDT.

`PollForNotifications`

Memungkinkan rangkaian tes untuk memeriksa notifikasi dari IDT.

`GetContextValue` dan `GetContextString`

Memungkinkan rangkaian tes untuk mengambil nilai-nilai dari konteks IDT. Untuk informasi selengkapnya, lihat Gunakan konteks IDT.

`SendResult`

Memungkinkan rangkaian tes untuk melaporkan hasil uji kasus ke IDT. Perintah ini harus dipanggil pada akhir setiap uji kasus di rangkaian tes.

Interaksi host

Perintah berikut memungkinkan rangkaian tes Anda untuk berkomunikasi dengan mesin host.

`PollForNotifications`

Memungkinkan rangkaian tes untuk memeriksa notifikasi dari IDT.

`GetContextValue` dan `GetContextString`

Memungkinkan rangkaian tes untuk mengambil nilai-nilai dari konteks IDT. Untuk informasi selengkapnya, lihat Gunakan konteks IDT.

`ExecuteOnHost`

Memungkinkan rangkaian tes untuk menjalankan perintah pada mesin lokal dan memungkinkan IDT untuk mengelola siklus hidup executable uji kasus.

Aktifkan perintah IDT CLI

Perintah `run-suite` IDT CLI menyediakan beberapa pilihan yang membiarkan test runner untuk mengkustomisasi pelaksanaan tes. Untuk memungkinkan test runner menggunakan opsi ini untuk menjalankan rangkaian tes kustom Anda, Anda menerapkan dukungan untuk IDT CLI. Jika Anda tidak menerapkan dukungan, test runner masih akan dapat menjalankan tes, tetapi beberapa opsi CLI tidak akan berfungsi dengan benar. Untuk memberikan pengalaman pelanggan yang ideal, kami merekomendasikan agar Anda menerapkan dukungan untuk argumen berikut untuk perintah `run-suite` dalam IDT CLI:

`timeout-multiplier`

Menentukan nilai yang lebih besar dari 1,0 yang akan diterapkan pada semua batas waktu saat menjalankan tes.

Test runner dapat menggunakan argumen ini untuk meningkatkan batas waktu untuk uji kasus yang ingin dijalankannya. Ketika test runner menentukan argumen ini pada perintah `run-suite`, IDT akan menggunakannya untuk menghitung nilai variabel lingkungan IDT_TEST_TIMEOUT dan menetapkan kolom `config.timeoutMultiplier` dalam konteks IDT. Untuk mendukung argumen ini, Anda harus melakukan hal berikut:

- Alih-alih langsung menggunakan nilai batas waktu dari file `test.json`, baca variabel lingkungan IDT_TEST_TIMEOUT untuk mendapatkan nilai batas waktu yang dihitung dengan benar.
- Ambil nilai `config.timeoutMultiplier` dari konteks IDT dan terapkan ia pada batas waktu yang panjang.

Untuk informasi selengkapnya tentang keluar lebih awal karena peristiwa habis waktu, lihat Tentukan perilaku keluar.

`stop-on-first-failure`

Tentukan bahwa IDT harus berhenti menjalankan semua tes jika menemui kegagalan.

Ketika test runner menentukan argumen ini pada perintah `run-suite`, IDT akan berhenti menjalankan pengujian tersebut segera setelah menemui kegagalan. Namun, jika uji kasus

berjalan secara paralel, hal ini dapat menyebabkan hasil yang tidak terduga. Untuk menerapkan dukungan, pastikan bahwa jika IDT menemui peristiwa ini, logika pengujian Anda akan menginstruksikan semua uji kasus yang sedang berjalan untuk berhenti, membersihkan sumber daya sementara, dan melaporkan hasil tes ke IDT. Untuk informasi selengkapnya tentang keluar lebih awal karena menemui kegagalan, lihat Tentukan perilaku keluar.

`group-id` dan `test-id`

Menentukan bahwa IDT harus menjalankan hanya grup uji atau uji kasus yang dipilih.

Test runner dapat menggunakan argumen ini dengan perintah `run-suite` untuk menentukan perilaku eksekusi tes berikut:

- Jalankan semua tes di dalam grup uji yang ditentukan.
- Jalankan pilihan tes dari dalam grup uji tertentu.

Untuk mendukung argumen ini, state machine untuk rangkaian tes Anda harus menyertakan serangkaian keadaan `RunTask` dan `Choice` tertentu pada state machine Anda. Jika Anda tidak menggunakan state machine kustom, maka state machine IDT default akan meliputi keadaan yang diperlukan untuk Anda dan Anda tidak perlu melakukan tindakan tambahan. Namun, jika Anda menggunakan state machine kustom, gunakan Contoh mesin status: Jalankan grup uji yang dipilih pengguna sebagai contoh untuk menambahkan keadaan yang diperlukan dalam state machine Anda.

Untuk informasi selengkapnya tentang perintah IDT CLI, lihat Debug dan jalankan rangkaian tes kustom.

Menulis log peristiwa

Saat tes berjalan, Anda mengirim data ke `stdout` dan `stderr` untuk menuliskan log peristiwa dan pesan kesalahan pada konsol. Untuk informasi lebih lanjut tentang format pesan konsol, lihat Format pesan konsol.

Ketika IDT selesai menjalankan rangkaian tes tersebut, informasi ini juga tersedia di file `test_manager.log` yang terletak di *`<devicetester-extract-location>`*`/results/`*`<execution-id>`*`/logs`.

Anda dapat mengonfigurasi setiap uji kasus untuk menuliskan log dari pengujiannya yang dijalankan, termasuk log dari perangkat yang diuji, ke file *`<group-id>_<test-id>`* yang terletak di *`<device-tester-extract-location>`*`/results/`*`execution-id`*`/logs`. Untuk melakukan ini, ambil path ke berkas log dari konteks IDT dengan kueri `testData.logFilePath`, buat file di path itu, dan

tuliskan konten yang Anda inginkan padanya. IDT secara otomatis memperbarui jalur berdasarkan uji kasus yang berjalan. Jika Anda memilih untuk tidak membuat file log untuk uji kasus, maka tidak ada file yang akan dibuat untuk uji kasus itu.

Anda juga dapat mengatur executable teks Anda untuk membuat berkas log tambahan yang diperlukan dalam folder *<device-tester-extract-location>*`/logs`. Kami menyarankan Anda untuk menentukan prefiks yang unik untuk nama file log sehingga file Anda tidak akan ditimpa.

Laporkan hasil ke IDT

IDT menuliskan hasil tes ke file `awsiotdevicetester_report.xml` dan *suite-name*`_report.xml`. File laporan ini terletak di *<device-tester-extract-location>*`/results/`*<execution-id>*`/`. Kedua laporan tersebut menangkap hasil dari eksekusi rangkaian tes. Untuk informasi selengkapnya tentang skema yang menggunakan IDT untuk laporan ini, lihat Tinjau hasil tes IDT dan log

Untuk mengisi konten file *suite-name*`_report.xml`, Anda harus menggunakan perintah `SendResult` untuk melaporkan hasil tes ke IDT sebelum eksekusi tes itu selesai. Jika IDT tidak dapat menemukan hasil tes, ia akan mengeluarkan kesalahan untuk uji kasus tersebut. Kutipan Python berikut menunjukkan perintah yang akan mengirimkan hasil tes ke IDT:

```
request-variable = SendResultRequest(TestResult(result))
client.send_result(request-variable)
```

Jika Anda tidak melaporkan hasil melalui API, IDT akan mencari hasil tes di folder artefak tes. Path ke folder ini disimpan dalam `testData.testArtifactsPath` yang disimpan dalam konteks IDT. Dalam folder ini, IDT menggunakan file XML yang diurutkan menurut abjad pertama yang ditempatkannya sebagai hasil tes.

Jika logika tes Anda menghasilkan hasil JUnit XML, Anda dapat menuliskan hasil tes itu ke file XML dalam folder artefak untuk langsung memberikan hasil ke IDT dan bukan mem-parsing hasilnya dan kemudian menggunakan API untuk mengirimkannya ke IDT.

Jika Anda menggunakan metode ini, pastikan bahwa logika pengujian Anda secara akurat merangkum hasil pengujian dan memformat file hasil Anda dalam format yang sama seperti file *suite-name*`_report.xml`. IDT tidak melakukan validasi data yang Anda berikan, dengan pengecualian berikut:

- IDT mengabaikan semua properti dari tanda `testsuites`. Sebaliknya, IDT menghitung properti tag dari hasil grup pengujian lainnya yang dilaporkan.

- Setidaknya satu `testsuite` tag harus ada dalam `testsuites`.

Karena IDT menggunakan folder artefak yang sama untuk semua uji kasus dan tidak menghapus file hasil antara pengujian yang berjalan, metode ini mungkin juga akan menyebabkan pelaporan yang salah jika IDT membaca file yang salah. Kami menyarankan Anda menggunakan nama yang sama untuk file hasil XML yang dihasilkan di semua uji kasus untuk menimpa hasil untuk setiap uji kasus dan pastikan bahwa hasil yang benar tersedia untuk digunakan oleh IDT. Meskipun Anda dapat menggunakan pendekatan campuran untuk pelaporan di rangkaian pengujian Anda, yaitu menggunakan file hasil XML untuk beberapa uji kasus dan mengirimkan hasil melalui API untuk uji kasus lainnya, kami tidak merekomendasikan pendekatan ini.

Tentukan perilaku keluar

Konfigurasikan executable teks Anda agar selalu keluar dengan kode keluar 0, meskipun uji kasus melaporkan kegagalan atau hasil kesalahan. Gunakan kode keluar bukan nol hanya untuk menunjukkan bahwa suatu uji kasus tidak berjalan atau jika executable uji kasus tidak dapat menyampaikan hasil apapun ke IDT. Ketika IDT menerima kode keluar bukan nol, IDT akan menandai uji kasus tersebut telah mengalami kesalahan yang mencegahnya berjalan.

IDT mungkin meminta atau mengharapkan uji kasus untuk berhenti berjalan sebelum selesai dalam peristiwa berikut. Gunakan informasi ini untuk mengonfigurasi executable uji kasus untuk mendeteksi setiap peristiwa ini dari uji kasus:

Waktu habis

Terjadi ketika uji kasus berjalan lebih lama daripada nilai batas waktu yang ditentukan dalam file `test.json`. Jika test runner menggunakan argumen `timeout-multiplier` untuk menentukan pengali batas waktu, IDT akan menghitung nilai batas waktu dengan pengali tersebut.

Untuk mendeteksi peristiwa ini, gunakan variabel lingkungan IDT_TEST_TIMEOUT. Ketika test runner meluncurkan tes, IDT akan menetapkan nilai variabel lingkungan IDT_TEST_TIMEOUT pada nilai batas waktu yang dihitung (dalam detik) dan melewati variabel pada executable uji kasus. Anda dapat membaca nilai variabel untuk menetapkan penghitung waktu yang sesuai.

Menginterupsi

Terjadi ketika test runner menginterupsi IDT. Misalnya, dengan menekan Ctrl+C.

Karena terminal menyebarkan sinyal ke semua proses anak, Anda cukup mengonfigurasi bagian yang menangani sinyal dalam uji kasus Anda untuk mendeteksi sinyal yang terinterupsi.

Atau, Anda dapat secara berkala mengumpulkan API untuk memeriksa nilai boolean `CancellationRequested` di respons API `PollForNotifications`. Ketika IDT menerima sinyal terinterupsi, ia akan menetapkan nilai boolean `CancellationRequested` untuk `true`.

Berhenti pada kegagalan pertama

Terjadi ketika uji kasus yang berjalan secara paralel dengan uji kasus gagal dan test runner menggunakan argumen `stop-on-first-failure` untuk menentukan bahwa IDT harus berhenti ketika menemui kegagalan apa pun.

Untuk mendeteksi peristiwa ini, Anda dapat secara berkala mengumpulkan API untuk memeriksa nilai boolean `CancellationRequested` di respons API `PollForNotifications`. Ketika IDT menemui kegagalan dan dikonfigurasi untuk berhenti pada kegagalan pertama, tetapkan nilai boolean `CancellationRequested` untuk `true`.

Ketika salah satu peristiwa ini terjadi, IDT akan menunggu selama 5 menit untuk setiap uji kasus yang sedang berjalan saat ini untuk menyelesaikan prosesnya. Jika semua uji kasus yang berjalan tidak keluar dalam waktu 5 menit, IDT akan memaksa masing-masing proses untuk berhenti. Jika IDT belum menerima hasil tes sebelum proses berakhir, ia akan menandai uji kasus telah habis waktu. Sebagai praktik terbaik, Anda harus memastikan bahwa uji kasus Anda melakukan tindakan berikut ketika menghadapi salah satu peristiwa berikut:

1. Berhenti menjalankan logika uji normal.
2. Bersihkan sumber daya sementara apa pun, seperti uji artefak pada perangkat yang sedang diuji.
3. Laporkan hasil tes ke IDT, seperti kegagalan uji atau kesalahan.
4. Keluar

Gunakan konteks IDT

Ketika IDT menjalankan rangkaian tes, rangkaian tes tersebut dapat mengakses serangkaian data yang dapat digunakan untuk menentukan bagaimana setiap tes akan berjalan. Data ini disebut konteks IDT. Sebagai contoh, konfigurasi data pengguna yang disediakan oleh test runner di file `userdata.json` tersedia pada rangkaian tes dalam konteks IDT.

Konteks IDT dapat dianggap sebagai dokumen JSON hanya-baca. Rangkaian uji dapat mengambil data dari dan menuliskan data ke konteks tersebut dengan menggunakan jenis data JSON standar seperti objek, rangkaian, angka, dan sebagainya.

Skema konteks

Konteks state machine menggunakan format berikut:

```
{
    "config": {
        <config-json-content>
        "timeoutMultiplier": timeout-multiplier
    },
    "device": {
        <device-json-device-element>
    },
    "devicePool": {
        <device-json-pool-element>
    },
    "resource": {
        "devices": [
            {
                <resource-json-device-element>
                "name": "<resource-name>"
            }
        ]
    },
    "testData": {
        "awsCredentials": {
            "awsAccessKeyId": "<access-key-id>",
            "awsSecretAccessKey": "<secret-access-key>",
            "awsSessionToken": "<session-token>"
        },
        "logFilePath": "/path/to/log/file"
    },
    "userData": {
        <userdata-json-content>
    }
}
```

`config`

Informasi dari `config.json` file. Kolom `config` juga berisi kolom tambahan berikut:

`config.timeoutMultiplier`

Pengganda untuk setiap nilai batas waktu yang digunakan oleh rangkaian tes. Nilai ini ditentukan oleh test runner dari IDT CLI. Nilai default-nya adalah 1.

`device`

Informasi tentang kolam perangkat yang dipilih untuk uji coba. Informasi ini setara dengan elemen rangkaian `devices` dalam file `device.json` untuk perangkat yang dipilih.

`devicePool`

Informasi tentang kolam perangkat yang dipilih untuk uji coba. Informasi ini setara dengan elemen rangkaian kolam perangkat tingkat ata yang ditentukan di file `device.json` untuk kolam perangkat yang dipilih.

`resource`

Informasi tentang perangkat sumber daya dari file `resource.json`.

`resource.devices`

Informasi ini setara dengan rangkaian `devices` yang ditentukan dalam file `resource.json`. Setiap elemen `devices` mencakup kolom tambahan berikut:

`resource.device.name`

Nama sumber daya. Nilai ini diatur ke nilai `requiredResource.name` pada file `test.json`.

`testData.awsCredentials`

Kredensial AWS yang digunakan oleh uji tersebut untuk terhubung ke cloud AWS. Informasi ini diperoleh dari file `config.json`.

`testData.logFilePath`

Path ke file log di mana uji kasus menuliskan pesan log. Rangkaian tes membuat file ini jika tidak ada.

`userData`

Informasi yang diberikan oleh test runner di file `userdata.json`.

Akses data dalam konteks

Anda dapat melakukan kueri atas konteks tersebut dengan menggunakan notasi JSONPath dari file JSON Anda dan dari executable teks Anda dengan API `GetContextValue` dan

`GetContextString`. Sintaks untuk string JSONPath untuk mengakses konteks IDT bervariasi sebagai berikut:

- Pada `suite.json` dan `test.json`, Anda menggunakan `{{query}}`. Artinya, jangan gunakan elemen root `$.` untuk memulai ekspresi Anda.
- Pada `statemachine.json`, Anda menggunakan `{{$.query}}`.
- Dalam perintah API, Anda menggunakan *`query`* atau `{{$.query}}`, tergantung pada perintahnya. Untuk informasi lebih lanjut, lihat dokumentasi sebaris in SDK.

Tabel berikut menjelaskan operator dalam ekspresi JSONPath yang khas:

Operator	Description
`$`	The root element. Because the top-level context value for IDT is an object, you will typically use `$.` to start your queries.
`.childName`	Accesses the child element with name `childName` from an object. If applied to an array, yields a new array with this operator applied to each element. The element name is case sensitive. For example, the query to access the `awsRegion` value in the `config` object is `$ .config.awsRegion` .
`[start:end]`	Filters elements from an array, retrieving items beginning from the `start` index and going up to the `end` index, both inclusive.
`[index1, index2,..., indexN]`	Filters elements from an array, retrieving items from only the specified indices.
`[?(expr)]`	Filters elements from an array using the `expr` expression. This expression must evaluate to a boolean value.

Untuk membuat ekspresi filter, gunakan sintaks berikut:

```
<jsonpath> | <value> operator <jsonpath> | <value>
```

Dalam sintaks ini:

- `jsonpath` adalah JSONPath yang menggunakan sintaks JSON standar.
- `value` adalah setiap nilai kustom yang menggunakan sintaks JSON standar.
- `operator` adalah salah satu dari operator berikut ini:
  - < (Kurang dari)
  - <= (Kurang dari atau sama dengan)
  - == (Sama dengan)

    Jika JSONPath atau nilai dalam ekspresi Anda adalah rangkaian, boolean, atau nilai objek, maka ini adalah satu-satunya operator biner yang didukung yang dapat Anda gunakan.
  - >= (Lebih besar dari atau sama dengan)
  - > (Lebih besar dari)
  - =~ (Kecocokan ekspresi reguler). Untuk menggunakan operator ini dalam ekspresi filter, JSONPath atau nilai di sisi kiri ekspresi Anda harus dievaluasi pada string dan sisi kanannya harus berupa nilai pola yang mengikuti sintaks RE2.

Anda dapat menggunakan kueri JSONPath dalam bentuk {{*query*}} sebagai string placeholder di kolom `args` dan `environmentVariables` pada file `test.json` dan pada kolom `environmentVariables` di file `suite.json`. IDT melakukan pencarian konteks dan mengisi kolom dengan nilai kueri yang dievaluasi. Misalnya, di file `suite.json`, Anda dapat menggunakan string placeholder untuk menentukan nilai variabel lingkungan yang berubah dengan setiap uji kasus dan IDT akan mengisi variabel lingkungan dengan nilai yang benar untuk setiap uji kasus. Namun, ketika Anda menggunakan string placeholder di file `test.json` dan `suite.json`, pertimbangan berikut berlaku untuk kueri Anda:

- Anda harus menuliskan setiap kejadian kunci `devicePool` dalam kueri Anda semua dengan huruf kecil. Artinya, gunakan `devicepool` sebagai gantinya
- Untuk rangkaian, Anda hanya dapat menggunakan rangkaian string. Selain itu, rangkaian menggunakan format `item1, item2,...,itemN` non-standar. Jika rangkaian tersebut hanya berisi satu elemen, maka rangkaian itu akan diserialkan sebagai `item`, sehingga menjadikannya tidak dapat dibedakan dari kolom string.
- Anda tidak dapat menggunakan placeholder untuk mengambil objek dari konteks.

Karena pertimbangan ini, kami merekomendasikan bahwa bila memungkinkan, Anda menggunakan API untuk mengakses konteks dalam logika pengujian Anda dan bukan string placeholder di file `test.json` dan `suite.json`. Namun, dalam beberapa kasus mungkin lebih nyaman untuk menggunakan placeholder JSONPath untuk mengambil string tunggal untuk ditetapkan sebagai variabel lingkungan.

Mengonfigurasi pengaturan untuk test runner

Untuk menjalankan rangkaian tes kustom, test runner harus mengonfigurasi pengaturannya berdasarkan rangkaia tes yang ingin dijalankannya. Pengaturan ditentukan berdasarkan templat file konfigurasi JSON yang terletak di folder *<device-tester-extract-location>*`/configs/`. Jika diperlukan, test runner juga harus mengatur kredensial AWS yang akan digunakan oleh IDT untuk terhubung ke cloud AWS.

Sebagai penyusun tes, Anda perlu mengonfigurasi file-file ini untuk men-debug rangkaian tes. Anda harus memberikan petunjuk kepada test runner agar dapat mengonfigurasi pengaturan berikut yang diperlukan untuk menjalankan rangkaian tes Anda.

Konfigurasikan device.json

File `device.json` berisi informasi tentang perangkat tempat uji dijalankan (misalnya, alamat IP, informasi login, sistem operasi, dan arsitektur CPU).

Test runner dapat memberikan informasi ini dengan menggunakan file `device.json` templat berikut yang terletak di folder *<device-tester-extract-location>*`/configs/`.

```
[
    {
        "id": "<pool-id>",
        "sku": "<pool-sku>",
        "features": [
            {
                "name": "<feature-name>",
                "value": "<feature-value>",
                "configs": [
                    {
                        "name": "<config-name>",
                        "value": "<config-value>"
                    }
                ],
```

```
            }
        ],
        "devices": [
            {
                "id": "<device-id>",
                "connectivity": {
                    "protocol": "ssh | uart | docker",
                    // ssh
                    "ip": "<ip-address>",
                    "port": <port-number>,
                    "auth": {
                        "method": "pki | password",
                        "credentials": {
                            "user": "<user-name>",
                            // pki
                            "privKeyPath": "/path/to/private/key",

                            // password
                            "password": "<password>",
                        }
                    },

                    // uart
                    "serialPort": "<serial-port>",

                    // docker
                    "containerId": "<container-id>",
                    "containerUser": "<container-user-name>",
                }
            }
        ]
    }
]
```

Semua kolom yang berisi nilai wajib diisi seperti yang dijelaskan di sini:

`id`

ID alfanumerik yang ditetapkan pengguna secara unik mengidentifikasi kumpulan perangkat yang disebut kolam perangkat. Perangkat yang termasuk dalam suatu kolam harus memiliki perangkat keras yang identik. Ketika Anda menjalankan serangkaian tes, perangkat di kolam tersebut digunakan untuk memparalelkan beban kerja. Beberapa perangkat digunakan untuk menjalankan tes yang berbeda.

`sku`

Nilai alfanumerik secara unik mengidentifikasi perangkat yang sedang diuji. SKU digunakan untuk melacak perangkat yang berkualitas.

> **Note**
>
> Jika Anda ingin mencantumkan forum Anda di Katalog perangkat AWS Partner, SKU yang Anda tentukan di sini harus sesuai dengan SKU yang Anda gunakan dalam proses pencantuman itu.

`features`

Tidak wajib. Rangkaian yang berisi fitur perangkat yang didukung. Fitur perangkat adalah nilai-nilai yang ditetapkan pengguna yang Anda konfigurasikan di rangkaian tes Anda. Anda harus memberikan informasi kepada test runner tentang nama fitur dan nilai-nilai yang akan disertakan dalam file `device.json`. Misalnya, jika Anda ingin menguji perangkat yang berfungsi sebagai server MQTT untuk perangkat lain, maka Anda dapat mengonfigurasi logika uji Anda untuk memvalidasi tingkat tertentu yang didukung untuk fitur bernama `MQTT_QOS`. Test runner memberikan nama fitur ini dan menetapkan nilai fitur ke tingkat QOS yang didukung oleh perangkatnya. Anda dapat mengambil informasi yang diberikan dari konteks IDT dengan kueri `devicePool.features`, atau dari konteks state machine dengan kueri `pool.features`.

`features.name`

Nama fitur.

`features.value`

Nilai fitur yang didukung.

`features.configs`

Pengaturan konfigurasi, jika diperlukan, untuk fitur.

`features.config.name`

Nama pengaturan konfigurasi.

`features.config.value`

Nilai pengaturan yang didukung.

`devices`

Rangkaian perangkat di kolam yang akan diuji. Setidaknya diperlukan satu perangkat.

`devices.id`

Pengenal unik yang ditetapkan pengguna untuk perangkat yang sedang diuji.

`connectivity.protocol`

Protokol komunikasi yang digunakan untuk berkomunikasi dengan perangkat ini. Setiap perangkat di kolam harus menggunakan protokol yang sama.

Saat ini, satu-satunya nilai yang didukung adalah `ssh` dan `uart` untuk perangkat fisik, dan `docker` untuk kontainer Docker.

`connectivity.ip`

Alamat IP perangkat yang sedang diuji.

Properti ini hanya berlaku jika `connectivity.protocol` diatur ke `ssh`.

`connectivity.port`

Tidak wajib. Jumlah port yang akan digunakan untuk koneksi SSH.

Nilai default-nya adalah 22.

Properti ini hanya berlaku jika `connectivity.protocol` diatur ke `ssh`.

`connectivity.auth`

Informasi autentikasi untuk koneksi tersebut.

Properti ini hanya berlaku jika `connectivity.protocol` diatur ke `ssh`.

`connectivity.auth.method`

Metode autentikasi yang digunakan untuk mengakses perangkat melalui protokol konektivitas yang diberikan.

Nilai yang didukung adalah:

- `pki`
- `password`

`connectivity.auth.credentials`

Kredensial yang digunakan untuk autentikasi.

`connectivity.auth.credentials.password`

Kata sandi yang digunakan untuk masuk ke perangkat yang sedang diuji.

Nilai ini hanya berlaku jika `connectivity.auth.method` diatur ke `password`.

`connectivity.auth.credentials.privKeyPath`

Jalur lengkap ke kunci privat yang digunakan untuk masuk ke perangkat yang sedang diuji.

Nilai ini hanya berlaku jika `connectivity.auth.method` diatur ke `pki`.

`connectivity.auth.credentials.user`

Nama pengguna untuk masuk ke perangkat yang sedang diuji.

`connectivity.serialPort`

Tidak wajib. Port serial tempat perangkat itu terhubung.

Properti ini hanya berlaku jika `connectivity.protocol` diatur ke `uart`.

`connectivity.containerId`

ID kontainer atau nama kontainer Docker yang sedang diuji.

Properti ini hanya berlaku jika `connectivity.protocol` diatur ke `docker`.

`connectivity.containerUser`

Tidak wajib. Nama pengguna untuk pengguna di dalam kontainer. Nilai default adalah pengguna yang disediakan di Dockerfile.

Nilai default-nya adalah 22.

Properti ini hanya berlaku jika `connectivity.protocol` diatur ke `docker`.

> **Note**
>
> Untuk memeriksa apakah test runner mengonfigurasi koneksi perangkat yang salah untuk suatu pengujian, Anda dapat mengambil

`pool.Devices[0].Connectivity.Protocol` dari konteks state machine dan membandingkannya dengan nilai yang diharapkan pada keadaan `Choice`. Jika protokol yang salah digunakan, cetak pesan dengan menggunakan keadaan `LogMessage` dan beralihlah ke keadaan `Fail`.
Atau, Anda dapat menggunakan kode penanganan kesalahan untuk melaporkan kegagalan pengujian untuk jenis perangkat yang salah.

(Opsional) Konfigurasikan userdata.json

File `userdata.json` berisi informasi tambahan yang diperlukan oleh rangkaian tes tetapi tidak ditentukan dalam file `device.json`. Format file ini tergantung pada file `userdata_scheme.json` yang ditentukan dalam rangkaian uji tersebut. Jika Anda seorang penyusun tes, pastikan Anda memberikan informasi ini kepada pengguna yang akan menjalankan rangkaian tes yang Anda susun.

(Opsional) Konfigurasikan resource.json

File `resource.json` berisi informasi tentang perangkat apa pun yang akan digunakan sebagai perangkat sumber daya. Perangkat sumber daya adalah perangkat yang diperlukan untuk menguji kemampuan tertentu dari perangkat yang diuji. Misalnya, untuk menguji kemampuan Bluetooth perangkat, Anda mungkin menggunakan perangkat sumber daya untuk menguji apakah perangkat Anda dapat berhasil tersambung. Perangkat sumber daya bersifat opsional, dan Anda dapat memerlukan perangkat sumber daya sebanyak yang Anda butuhkan. Sebagai penyusun tes, Anda menggunakan file test.json untuk menentukan fitur perangkat sumber daya yang diperlukan untuk tes. Test runner kemudian akan menggunakan file `resource.json` untuk menyediakan kolam perangkat sumber daya yang memiliki fitur yang diperlukan. Pastikan Anda memberikan informasi ini kepada pengguna yang akan menjalankan rangkaian tes yang Anda tulis.

Test runner dapat memberikan informasi ini dengan menggunakan file `resource.json` templat berikut yang terletak di folder *`<device-tester-extract-location>`*`/configs/`.

```
[
    {
        "id": "<pool-id>",
        "features": [
            {
                "name": "<feature-name>",
                "version": "<feature-value>",
                "jobSlots": <job-slots>
```

```
            }
        ],
        "devices": [
            {
                "id": "<device-id>",
                "connectivity": {
                    "protocol": "ssh | uart | docker",
                    // ssh
                    "ip": "<ip-address>",
                    "port": <port-number>,
                    "auth": {
                        "method": "pki | password",
                        "credentials": {
                            "user": "<user-name>",
                            // pki
                            "privKeyPath": "/path/to/private/key",

                            // password
                            "password": "<password>",
                        }
                    },

                    // uart
                    "serialPort": "<serial-port>",

                    // docker
                    "containerId": "<container-id>",
                    "containerUser": "<container-user-name>",
                }
            }
        ]
    }
]
```

Semua kolom yang berisi nilai wajib diisi seperti yang dijelaskan di sini:

`id`

ID alfanumerik yang ditetapkan pengguna secara unik mengidentifikasi kumpulan perangkat yang disebut kolam perangkat. Perangkat yang termasuk dalam suatu kolam harus memiliki perangkat keras yang identik. Ketika Anda menjalankan serangkaian tes, perangkat di kolam tersebut digunakan untuk memparalelkan beban kerja. Beberapa perangkat digunakan untuk menjalankan tes yang berbeda.

`features`

Tidak wajib. Rangkaian yang berisi fitur perangkat yang didukung. Informasi yang diperlukan dalam kolom ini ditentukan dalam file test.json di rangkaian tes dan menentukan tes mana yang akan dijalankan dan bagaimana menjalankan tes tersebut. Jika rangkaian tes tidak memerlukan fitur apa pun, kolom ini tidak wajib diisi.

`features.name`

Nama fitur.

`features.version`

Versi fitur.

`features.jobSlots`

Pengaturan untuk menunjukkan berapa banyak tes yang dapat secara bersamaan menggunakan perangkat. Nilai default-nya adalah 1.

`devices`

Rangkaian perangkat di kolam yang akan diuji. Setidaknya diperlukan satu perangkat.

`devices.id`

Pengenal unik yang ditetapkan pengguna untuk perangkat yang sedang diuji.

`connectivity.protocol`

Protokol komunikasi yang digunakan untuk berkomunikasi dengan perangkat ini. Setiap perangkat di kolam harus menggunakan protokol yang sama.

Saat ini, satu-satunya nilai yang didukung adalah `ssh` dan `uart` untuk perangkat fisik, dan `docker` untuk kontainer Docker.

`connectivity.ip`

Alamat IP perangkat yang sedang diuji.

Properti ini hanya berlaku jika `connectivity.protocol` diatur ke `ssh`.

`connectivity.port`

Tidak wajib. Jumlah port yang akan digunakan untuk koneksi SSH.

Nilai default-nya adalah 22.

Properti ini hanya berlaku jika `connectivity.protocol` diatur ke `ssh`.

`connectivity.auth`

Informasi autentikasi untuk koneksi tersebut.

Properti ini hanya berlaku jika `connectivity.protocol` diatur ke `ssh`.

`connectivity.auth.method`

Metode autentikasi yang digunakan untuk mengakses perangkat melalui protokol konektivitas yang diberikan.

Nilai yang didukung adalah:

- `pki`
- `password`

`connectivity.auth.credentials`

Kredensial yang digunakan untuk autentikasi.

`connectivity.auth.credentials.password`

Kata sandi yang digunakan untuk masuk ke perangkat yang sedang diuji.

Nilai ini hanya berlaku jika `connectivity.auth.method` diatur ke `password`.

`connectivity.auth.credentials.privKeyPath`

Jalur lengkap ke kunci privat yang digunakan untuk masuk ke perangkat yang sedang diuji.

Nilai ini hanya berlaku jika `connectivity.auth.method` diatur ke `pki`.

`connectivity.auth.credentials.user`

Nama pengguna untuk masuk ke perangkat yang sedang diuji.

`connectivity.serialPort`

Tidak wajib. Port serial tempat perangkat itu terhubung.

Properti ini hanya berlaku jika `connectivity.protocol` diatur ke `uart`.

`connectivity.containerId`

ID kontainer atau nama kontainer Docker yang sedang diuji.

Properti ini hanya berlaku jika `connectivity.protocol` diatur ke `docker`.

`connectivity.containerUser`

Tidak wajib. Nama pengguna untuk pengguna di dalam kontainer. Nilai default adalah pengguna yang disediakan di Dockerfile.

Nilai default-nya adalah 22.

Properti ini hanya berlaku jika `connectivity.protocol` diatur ke `docker`.

(Opsional) Konfigurasikan config.json

File `config.json` berisi informasi konfigurasi untuk IDT. Biasanya,tes runner tidak perlu memodifikasi file ini kecuali untuk menyediakan AWS kredensial pengguna untuk IDT, dan opsional, sebuah AWS wilayah. Jika AWS kredensial dengan izin yang diperlukan disediakan AWS IoT Penguji Perangkat mengumpulkan dan mengirimkan metrik penggunaan ke AWS. Ini adalah fitur opt-in dan digunakan untuk meningkatkan fungsi IDT. Untuk informasi selengkapnya, lihat Metrik penggunaan IDT.

Test runner dapat mengonfigurasi kredensial AWS dengan salah satu cara berikut:

- File kredensial

  IDT menggunakan file kredensial yang sama sebagai AWS CLI. Untuk informasi selengkapnya, lihat File konfigurasi dan kredensial.

  Lokasi file kredensial itu bervariasi, tergantung pada sistem operasi yang Anda gunakan:

  - macOS, Linux: `~/.aws/credentials`
  - Windows: `C:\Users\UserName\.aws\credentials`
- Variabel lingkungan

  Variabel lingkungan adalah variabel yang dikelola oleh sistem operasi dan digunakan oleh perintah sistem. Variabel yang ditentukan selama sesi SSH tidak akan tersedia setelah sesi itu ditutup. IDT dapat menggunakan variabel lingkungan `AWS_ACCESS_KEY_ID` dan `AWS_SECRET_ACCESS_KEY` untuk menyimpan kredensial AWS

Untuk mengatur variabel ini di Linux, macOS, atau Unix, gunakan export:

```
export AWS_ACCESS_KEY_ID=<your_access_key_id>
export AWS_SECRET_ACCESS_KEY=<your_secret_access_key>
```

Untuk menetapkan variabel ini pada Windows, gunakan set:

```
set AWS_ACCESS_KEY_ID=<your_access_key_id>
set AWS_SECRET_ACCESS_KEY=<your_secret_access_key>
```

Untuk mengonfigurasi kredensial AWS untuk IDT, test runner akan mengedit bagian `auth` dalam file `config.json` yang terletak di *<device-tester-extract-location>*`/configs/`.

```
{
    "log": {
        "location": "logs"
    },
    "configFiles": {
        "root": "configs",
        "device": "configs/device.json"
    },
    "testPath": "tests",
    "reportPath": "results",
    "awsRegion": "<region>",
    "auth": {
        "method": "file | environment",
        "credentials": {
            "profile": "<profile-name>"
        }
    }
}
]
```

Semua kolom yang berisi nilai wajib diisi seperti yang dijelaskan di sini:

> **Note**
>
> Semua path dalam file ini didefinisikan relatif terhadap*<device-tester-extract-location>*.

`log.location`

Path ke folder log di *<device-tester-extract-location>*.

`configFiles.root`

Path ke folder yang berisi file konfigurasi.

`configFiles.device`

Jalur ke file `device.json`.

`testPath`

Path ke folder yang berisi rangkaian tes.

`reportPath`

Path ke folder yang akan berisi hasil tes setelah IDT menjalankan rangkaian tes.

`awsRegion`

Tidak wajib. Wilayah AWS yang akan digunakan oleh rangkaian uji. Jika tidak ditetapkan, rangkaian tes akan menggunakan wilayah default yang ditentukan dalam setiap rangkaian tes.

`auth.method`

Metode IDT yang digunakan untuk mengambil kredensial AWS. Nilai yang didukung adalah `file` untuk mengambil kredensial dari file kredensial, dan `environment` untuk mengambil klredensial dengan menggunakan variabel lingkungan.

`auth.credentials.profile`

Profil kredensial yang akan digunakan dari file kredensial. Properti ini hanya berlaku jika `auth.method` diatur ke `file`.

Debug dan jalankan rangkaian tes kustom

Setelah konfigurasi yang diperlukan diatur, IDT dapat menjalankan rangkaian tes Anda. Waktu aktif dari rangkaian tes penuh akan tergantung pada perangkat keras dan komposisi rangkaian tes. Untuk referensi, dibutuhkan waktu sekitar 30 menit untuk menyelesaikan rangkaian tes kualifikasi AWS IoT Greengrass pada 3B Raspberry Pi.

Ketika Anda menyusun rangkaian tes Anda, Anda dapat menggunakan IDT untuk menjalankan rangkaian tes dalam mode debug untuk memeriksa kode Anda sebelum Anda menjalankannya atau memberikannya kepada test runner.

Jalankan IDT dalam mode debug

Karena rangkaian tes tergantung pada IDT untuk berinteraksi dengan perangkat, menyediakan konteks, dan menerima hasil, Anda tidak bisa hanya men-debug rangkaian tes Anda di IDE tanpa berinteraksi dengan IDT. Untuk melakukannya, IDT CLI menyediakan perintah `debug-test-suite` yang memungkinkan Anda menjalankan IDT dalam mode debug. Jalankan perintah berikut untuk menampilkan opsi yang tersedia untuk `debug-test-suite`:

```
devicetester_[linux | mac | win_x86-64] debug-test-suite -h
```

Ketika Anda menjalankan IDT dalam mode debug, IDT tidak benar-benar meluncurkan rangkaian tes atau menjalankan state machine; sebaliknya, IDT berinteraksi dengan IDE Anda untuk menanggapi permintaan yang dibuat dari rangkaian tes yang berjalan di IDE dan mencetak log ke konsol. IDT tidak melakukan timeout dan menunggu untuk keluar hingga secara manual terinterupsi. Dalam mode debug, IDT juga tidak menjalankan state machine dan tidak akan menghasilkan file laporan. Untuk men-debug rangkaian tes Anda, Anda harus menggunakan IDE Anda untuk memberikan beberapa informasi yang biasanya diperoleh IDT dari file JSON konfigurasi. Pastikan Anda memberikan informasi berikut:

- Variabel lingkungan dan argumen untuk setiap tes. IDT tidak akan membaca informasi ini dari `test.json` atau `suite.json`.
- Argumen untuk memilih perangkat sumber daya. IDT tidak akan membaca informasi ini dari `test.json`.

Untuk men-debug rangkaian tes Anda, selesaikan langkah berikut:

1. Buat file konfigurasi pengaturan yang diperlukan untuk menjalankan rangkaian tes. Misalnya, jika rangkaian tes Anda memerlukan `device.json`, `resource.json`, dan `user data.json`, pastikan Anda mengonfigurasi semuanya sesuai kebutuhan.
2. Jalankan perintah berikut untuk menempatkan IDT dalam mode debug dan pilih perangkat yang diperlukan untuk menjalankan tes.

   ```
   devicetester_[linux | mac | win_x86-64] debug-test-suite [options]
   ```

   Setelah Anda menjalankan perintah ini, IDT akan menunggu permintaan dari rangkaian tes dan kemudian menanggapinya. IDT juga akan menghasilkan variabel lingkungan yang diperlukan untuk proses kasus untuk IDT Client SDK.

3. Dalam IDE Anda, gunakan konfigurasi `run` atau `debug` untuk melakukan hal berikut:
   a. Menetapkan nilai-nilai variabel lingkungan yang dihasilkan IDT.
   b. Tetapkan nilai dari setiap variabel lingkungan atau argumen yang Anda tentukan dalam file `test.json` dan `suite.json` Anda.
   c. Menetapkan breakpoint sesuai kebutuhan.
4. Menjalankan rangkaian tes di IDE Anda.

   Anda dapat men-debug dan kembali menjalankan rangkaain tes sebanyak mungkin yang diperlukan. IDT tidak melakukan timeout dalam mode debug.
5. Setelah Anda menyelesaikan debugging, interupsi IDT untuk keluar dari mode debug.

Perintah IDT CLI untuk menjalankan percobaan

Bagian berikut menjelaskan perintah IDT CLI:

IDT v4.0.0

`help`

Mendaftar informasi tentang perintah yang ditentukan.

`list-groups`

Mendaftar grup dalam rangkaian tes yang diberikan.

`list-suites`

Mendaftar rangkaian tes yang tersedia.

`list-supported-products`

Mencantumkan produk yang didukung untuk versi IDT Anda, dalam hal ini versi AWS IoT Greengrass, dan versi rangkaian uji kualifikasi AWS IoT Greengrass yang tersedia untuk versi IDT saat ini.

`list-test-cases`

Daftar uji kasus dalam grup uji yang diberikan. Opsi berikut didukung:

- `group-id`. Grup uji yang harus dicari. Opsi ini diperlukan dan harus menentukan satu grup.

`run-suite`

Menjalankan serangkaian tes pada kolam perangkat. Berikut ini adalah beberapa opsi yang umum digunakan:

- `suite-id`. Versi rangkaian tes yang akan dijalankan. Jika tidak ditentukan, IDT akan menggunakan versi terbaru dalam folder `tests`.
- `group-id`. Grup uji yang akan jalankan, sebagai daftar yang dipisahkan koma. Jika tidak ditentukan, IDT akan menjalankan semua grup uji di rangkaian tes.
- `test-id`. Uji kasus yang akan dijalankan, sebagai daftar yang dipisahkan koma. Ketika ditentukan, `group-id` harus menentukan satu grup.
- `pool-id`. Kolam perangkat yang akan diuji. Test runner harus menentukan kolam jika memiliki beberapa perangkat kolam yang ditentukan dalam file `device.json`.
- `timeout-multiplier`. Mengkonfigurasi IDT untuk mengubah batas waktu eksekusi tes yang ditentukan dalam file `test.json` untuk tes dengan pengganda yang ditetapkan pengguna.
- `stop-on-first-failure`. Mengkonfigurasi IDT untuk menghentikan eksekusi pada kegagalan pertama. Pilihan ini harus digunakan dengan `group-id` untuk men-debug grup uji yang ditentukan.
- `userdata`. Menetapkan file yang berisi informasi data pengguna yang diperlukan untuk menjalankan rangkaian tes. Hal ini hanya diperlukan jika `userdataRequired` diatur ke BETUL di file `suite.json`untuk rangkaian uji itu.

Untuk informasi lebih lanjut tentang opsi `run-suite`, gunakan opsi `help`:

```
devicetester_[linux | mac | win_x86-64] run-suite -h
```

`debug-test-suite`

Jalankan rangkaian tes dalam mode debug. Untuk informasi selengkapnya, lihat Jalankan IDT dalam mode debug.

Tinjau hasil tes IDT dan log

Bagian ini menjelaskan format di mana IDT menghasilkan log konsol dan laporan tes.

Format pesan konsol

AWS IoT Penguji Perangkat menggunakan format standar untuk mencetak pesan ke konsol ketika memulai rangkaian pengujian. Kutipan berikut menunjukkan contoh pesan konsol yang dihasilkan oleh IDT.

```
time="2000-01-02T03:04:05-07:00" level=info msg=Using suite: MyTestSuite_1.0.0
executionId=9a52f362-1227-11eb-86c9-8c8590419f30
```

Sebagian besar pesan konsol terdiri dari kolom berikut:

`time`

Sebuah cap waktu ISO 8601 penuh untuk peristiwa yang tercatat.

`level`

Tingkat pesan untuk peristiwa yang tercatat. Biasanya, tingkat pesan yang tercatat adalah salah satu dari `info`, `warn`, atau `error`. IDT mengeluarkan pesan `fatal` atau `panic` jika bertemu dengan peristiwa yang diharapkan yang menyebabkannya keluar lebih awal.

`msg`

Pesan yang dicatat.

`executionId`

Sebuah string ID yang unik untuk proses IDT saat ini. ID ini digunakan untuk membedakan antara IDT individual yang berjalan.

Pesan konsol yang dihasilkan dari rangkaian tes memberikan informasi tambahan tentang perangkat yang diuji berikut rangkaian uji, grup uji, dan uji kasus yang dijalankan oleh IDT. Kutipan berikut menunjukkan contoh pesan konsol yang dihasilkan dari rangkaian tes.

```
time="2000-01-02T03:04:05-07:00" level=info msg=Hello world! suiteId=MyTestSuite
groupId=myTestGroup testCaseId=myTestCase deviceId=my-device
executionId=9a52f362-1227-11eb-86c9-8c8590419f30
```

Bagian tertentu dari rangkaian tes pada pesan konsol berisi kolom-kolom berikut:

`suiteId`

Nama rangkaian tes yang sedang berjalan saat ini.

`groupId`

ID grup uji yang sedang berjalan saat ini.

`testCaseId`

ID uji kasus yang berjalan saat ini.

`deviceId`

ID dari perangkat yang diuji yang sedang digunakan oleh uji kasus saat ini.

Untuk mencetak ringkasan tes ke konsol tersebut ketika IDT selesai menjalankan pengujian, Anda harus menyertakan keadaan `Report` pada state machine Anda. Ringkasan uji berisi informasi tentang rangkaian uji, hasil uji untuk setiap grup yang dijalankan, dan lokasi log dan file laporan yang dihasilkan. Contoh berikut menunjukkan pesan ringkasan tes.

```
========== Test Summary ==========
Execution Time:     5m00s
Tests Completed:    4
Tests Passed:       3
Tests Failed:       1
Tests Skipped:      0
---------------------------------
Test Groups:
    GroupA:         PASSED
    GroupB:         FAILED
---------------------------------
Failed Tests:
    Group Name: GroupB
        Test Name: TestB1
            Reason: Something bad happened
---------------------------------
Path to IoT Device Tester Report: /path/to/awsiotdevicetester_report.xml
Path to Test Execution Logs: /path/to/logs
Path to Aggregated JUnit Report: /path/to/MyTestSuite_Report.xml
```

AWS IoT Skema laporan Penguji Perangkat

`awsiotdevicetester_report.xml` adalah laporan ditandatangani yang berisi informasi berikut:

- Versi IDT.

- Versi rangkaian tes.
- Tanda tangan laporan dan kunci yang digunakan untuk menandatangani laporan.
- SKU Perangkat dan nama kolam perangkat yang ditentukan dalam file `device.json`.
- Versi produk dan fitur perangkat yang diuji.
- Ringkasan agregat hasil tes. Informasi ini sama dengan yang terkandung dalam file *`suite-name`*`_report.xml`.

```
<apnreport>
    <awsiotdevicetesterversion>idt-version</awsiotdevicetesterversion>
    <testsuiteversion>test-suite-version</testsuiteversion>
    <signature>signature</signature>
    <keyname>keyname</keyname>
    <session>
        <testsession>execution-id</testsession>
        <starttime>start-time</starttime>
        <endtime>end-time</endtime>
    </session>
    <awsproduct>
        <name>product-name</name>
        <version>product-version</version>
        <features>
            <feature name="<feature-name>" value="supported | not-supported | <feature-
value>" type="optional | required"/>
        </features>
    </awsproduct>
    <device>
        <sku>device-sku</sku>
        <name>device-name</name>
        <features>
            <feature name="<feature-name>" value="<feature-value>"/>
        </features>
        <executionMethod>ssh | uart | docker</executionMethod>
    </device>
    <devenvironment>
        <os name="<os-name>"/>
    </devenvironment>
    <report>
        <suite-name-report-contents>
    </report>
</apnreport>
```

File `awsiotdevicetester_report.xml` berisi tanda `<awsproduct>` yang berisi informasi tentang produk yang sedang diuji dan fitur produk yang divalidasi setelah menjalankan serangkaian pengujian.

Atribut yang digunakan dalam tanda **`<awsproduct>`**

`name`

Nama produk yang sedang diuji.

`version`

Versi produk yang sedang diuji.

`features`

Fitur divalidasi. Fitur yang ditandai sebagai `required` diperlukan bagi rangkaian tes untuk memvalidasi perangkat. Potongan berikut menunjukkan bagaimana informasi ini muncul di file `awsiotdevicetester_report.xml`.

```
<feature name="ssh" value="supported" type="required"></feature>
```

Fitur yang ditandai sebagai `optional` tidak diperlukan untuk validasi. Potongan berikut menunjukkan fitur opsional.

```
<feature name="hsi" value="supported" type="optional"></feature>

<feature name="mqtt" value="not-supported" type="optional"></feature>
```

Skema laporan rangkaian uji

Laporan *`suite-name`*`_Result.xml` berada dalam format JUnitXML. Anda dapat mengintegrasikannya ke dalam platform integrasi dan deployment berkelanjutan seperti Jenkins, Bamboo, dan sebagainya. Laporan ini berisi ringkasan agregat hasil tes.

```
<testsuites name="<suite-name> results" time="<run-duration>" tests="<number-of-test>"
 failures="<number-of-tests>" skipped="<number-of-tests>" errors="<number-of-tests>"
 disabled="0">
    <testsuite name="<test-group-id>" package="" tests="<number-of-tests>"
 failures="<number-of-tests>" skipped="<number-of-tests>" errors="<number-of-tests>"
 disabled="0">
```

```
        <!--success-->
        <testcase classname="<classname>" name="<name>" time="<run-duration>"/>
        <!--failure-->
        <testcase classname="<classname>" name="<name>" time="<run-duration>">
            <failure type="<failure-type>">
                reason
            </failure>
        </testcase>
        <!--skipped-->
        <testcase classname="<classname>" name="<name>" time="<run-duration>">
            <skipped>
                reason
            </skipped>
        </testcase>
        <!--error-->
        <testcase classname="<classname>" name="<name>" time="<run-duration>">
            <error>
                reason
            </error>
        </testcase>
    </testsuite>
</testsuites>
```

Bagian laporan baik di `awsiotdevicetester_report.xml` maupun *`suite-name`*`_report.xml` mendaftar tes yang dijalankan dan hasilnya.

Tag XML pertama `<testsuites>` berisi ringkasan pelaksanaan tes. Misalnya:

```
<testsuites name="MyTestSuite results" time="2299" tests="28" failures="0" errors="0"
 disabled="0">
```

Atribut yang digunakan dalam tanda **`<testsuites>`**

`name`

Nama rangkaian tes.

`time`

Waktu, dalam hitungan detik, yang dibutuhkannya untuk menjalankan rangkaian tes.

`tests`

Jumlah tes yang dilaksanakan.

`failures`

Jumlah tes yang dijalankan, tetapi tidak lulus.

`errors`

Jumlah tes yang tidak dapat dilaksanakan oelh IDT.

`disabled`

Atribut ini tidak digunakan dan bisa diabaikan.

Jika pengujian gagal atau salah, Anda dapat mengidentifikasi pengujian yang gagal dengan meninjau tanda XML `<testsuites>`. Tag XML `<testsuite>` di dalam tag `<testsuites>` menunjukkan ringkasan hasil tes untuk grup uji. Misalnya:

```
<testsuite name="combination" package="" tests="1" failures="0" time="161" disabled="0"
 errors="0" skipped="0">
```

Format ini serupa dengan tanda `<testsuites>`, tetapi dengan atribut `skipped` yang tidak digunakan dan dapat diabaikan. Di dalam masing-masing tanda XML `<testsuite>`, terdapat tanda `<testcase>` untuk setiap tes yang dieksekusi untuk suatu grup uji. Misalnya:

```
<testcase classname="Security Test" name="IP Change Tests" attempts="1"></testcase>>
```

Atribut yang digunakan dalam tanda **`<testcase>`**

`name`

Nama tes.

`attempts`

Berapa kali IDT mengeksekusi uji kasus.

Ketika tes gagal atau kesalahan terjadi, tag `<failure>` atau `<error>` akan ditambahkan ke tag `<testcase>` dengan informasi untuk pemecahan masalah. Misalnya:

```
<testcase classname="mcu.Full_MQTT" name="MQTT_TestCase" attempts="1">
 <failure type="Failure">Reason for the test failure</failure>
 <error>Reason for the test execution error</error>
```

```
</testcase>
```

Metrik penggunaan IDT

Jika Anda menyediakan AWS kredensial dengan izin yang diperlukan, AWS IoT Penguji Perangkat mengumpulkan dan mengirimkan metrik penggunaan ke AWS. Ini adalah fitur opt-in dan digunakan untuk meningkatkan fungsi IDT. IDT mengumpulkan informasi seperti berikut:

- ID Akun AWS digunakan untuk menjalankan IDT
- Perintah IDT CLI yang digunakan untuk menjalankan tes
- Rangkaian tes yang dijalankan
- Rangkaian pengujian di folder *< device-tester-extract-location >*
- Jumlah perangkat yang dikonfigurasi dalam kolam perangkat
- Nama uji kasus dan waktu aktif
- Informasi hasil tes, seperti apakah tes berhasil dilalui, gagal, mengalami kesalahan, atau dilewati
- Fitur produk yang diuji
- Perilaku keluar IDT, seperti keluar tak terduga atau lebih awal

Semua informasi yang dikirimkan IDT juga dicatat pada file `metrics.log` di folder *<device-tester-extract-location>*`/results/`*<execution-id>*`/`. Anda dapat melihat file log untuk melihat informasi yang dikumpulkan ketika tes dijalankan. File ini dibuat hanya jika Anda memilih untuk mengumpulkan metrik penggunaan.

Untuk menonaktifkan pengumpulam metrik, Anda tidak perlu melakukan tindakan tambahan. Jangan simpan kredensial AWS Anda, dan jika Anda telah menyimpan kredensial AWS, jangan konfigurasikan file n `config.jso` untuk mengaksesnya.

Konfigurasikan kredensial AWS Anda

Jika Anda belum memiliki Akun AWS, Anda harus membuatnya. Jika Anda sudah memiliki Akun AWS, Anda hanya perlu mengonfigurasi izin yang diperlukan untuk akun Anda yang memungkinkan IDT mengirim metrik penggunaan ke AWS atas nama Anda.

Langkah 1: Buat Akun AWS

Pada langkah ini, buat dan konfigurasikan Akun AWS. Jika Anda sudah memiliki akun Akun AWS, lewati ke the section called “Langkah 2: Konfigurasikan izin untuk IDT”.

Mendaftar Akun AWS

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar Akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.
2. Ikuti petunjuk secara online.

   Anda akan diminta untuk menerima panggilan telepon dan memasukkan kode verifikasi pada keypad telepon sebagai bagian dari prosedur pendaftaran.

   Saat Anda mendaftar Akun AWS, Pengguna root akun AWS akan dibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya dalam akun. Sebagai praktik terbaik keamanan, tetapkan akses administratif ke pengguna administratif, dan hanya gunakan pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

AWS akan mengirimkan email konfirmasi kepada Anda setelah proses pendaftaran selesai. Anda dapat melihat aktivitas akun saat ini dan mengelola akun dengan mengunjungi https://aws.amazon.com/ dan memilih Akun Saya.

Membuat pengguna administratif

Setelah mendaftar Akun AWS, amankan Pengguna root akun AWS, aktifkan AWS IAM Identity Center, dan buat sebuah pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Mengamankan Pengguna root akun AWS Anda

1. Masuk ke AWS Management Console sebagai pemilik akun dengan memilih Pengguna root dan memasukkan alamat email Akun AWS Anda. Di halaman berikutnya, masukkan kata sandi Anda.

   Untuk bantuan masuk menggunakan pengguna root, lihat Masuk sebagai pengguna root dalam Panduan Pengguna AWS Sign-In.
2. Aktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda.

   Untuk petunjuknya, silakan lihat Mengaktifkan perangkat MFA virtual untuk pengguna root Akun AWS Anda (konsol) dalam Panduan Pengguna IAM.

Membuat pengguna administratif

1. Aktifkan Pusat Identitas IAM.

   Untuk mendapatkan petunjuk, silakan lihat Mengaktifkan AWS IAM Identity Center di Panduan Pengguna AWS IAM Identity Center.

2. Di Pusat Identitas IAM, berikan akses administratif ke sebuah pengguna administratif.

   Untuk mendapatkan tutorial tentang menggunakan Direktori Pusat Identitas IAM sebagai sumber identitas Anda, silakan lihat Mengonfigurasi akses pengguna dengan Direktori Pusat Identitas IAM default di Panduan Pengguna AWS IAM Identity Center.

Masuk sebagai pengguna administratif

- Untuk masuk dengan pengguna Pusat Identitas IAM, gunakan URL masuk yang dikirim ke alamat email Anda saat Anda membuat pengguna Pusat Identitas IAM.

  Untuk bantuan masuk menggunakan pengguna Pusat Identitas IAM, lihat Masuk ke portal akses AWS dalam Panduan Pengguna AWS Sign-In.

Langkah 2: Konfigurasikan izin untuk IDT

Pada langkah ini, konfigurasikan izin yang menggunakan IDT untuk menjalankan tes dan mengumpulkan data penggunaan IDT. Anda dapat menggunakan AWS Management Console atau AWS Command Line Interface (AWS CLI) untuk membuat kebijakan IAM dan pengguna untuk IDT, dan kemudian melampirkan kebijakan untuk pengguna.

- Untuk Mengkonfigurasi Izin untuk IDT (Konsol)
- Untuk Mengkonfigurasi Izin untuk IDT () AWS CLI

Untuk mengonfigurasi izin untuk IDT (konsol)

Ikuti langkah berikut untuk menggunakan konsol untuk mengonfigurasi izin untuk IDT untuk AWS IoT Greengrass.

1. Masuklah ke konsol IAM.
2. Buat kebijakan yang dikelola pelanggan yang memberikan izin untuk membuat peran dengan izin tertentu.

a. Pada panel navigasi, pilih Kebijakan, lalu pilih Buat kebijakan.

b. Pada tab JSON, ganti placeholder konten dengan kebijakan berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot-device-tester:SendMetrics"
            ],
            "Resource": "*"
        }
    ]
}
```

c. Pilih Berikutnya: Tanda.

d. Pilih Berikutnya: Tinjauan.

e. Untuk Nama, masukkan **`IDTUsageMetricsIAMPermissions`**. Di bawah Ringkasan, tinjau izin yang diberikan oleh kebijakan Anda.

f. Pilih Buat kebijakan.

3. Buatlah pengguna IAM dan lampirkan izin untuk pengguna.

a. Buat pengguna IAM. Ikuti langkah 1 hingga 5 di Membuat pengguna IAM (konsol) di Panduan Pengguna IAM. Jika Anda sudah membuat pengguna IAM, lewati ke langkah berikutnya.

b. Lampirkan izin untuk pengguna IAM Anda:

i. Pada halaman Setel izin, pilih Lampirkan kebijakan yang ada secara langsung.

ii. Cari kebijakan IDT UsageMetrics IAmPermissions yang Anda buat di langkah sebelumnya. Pilih kotak centang.

c. Pilih Selanjutnya: Menandai.

d. Pilih Berikutnya: Tinjauan untuk melihat ringkasan pilihan Anda.

e. Pilih Buat pengguna.

f. Untuk melihat access key pengguna (access key ID dan secret access key), pilih Tampilkan di samping setiap kata sandi dan kunci akses rahasia. Untuk menyimpan kunci akses, pilih

Download.csv lalu simpan file ke lokasi yang aman. Anda menggunakan informasi ini nanti untuk mengonfigurasi file kredensial AWS Anda.

Untuk mengonfigurasi izin untuk IDT (AWS CLI)

Ikuti langkah-langkah ini untuk menggunakan AWS CLI agar mengonfigurasi untuk IDT pada AWS IoT Greengrass. Jika Anda sudah mengonfigurasi izin di konsol, lewati ke the section called "Konfigurasikan perangkat Anda untuk menjalankan tes IDT" atau the section called "Opsional: Mengonfigurasi kontainer Docker".

1. Pada komputer Anda, instal dan konfigurasikan AWS CLI jika ia belum dipasang. Ikuti langkah-langkah di Menginstal AWS CLI di Panduan Pengguna AWS Command Line Interface.

   > **Note**
   >
   > AWS CLI adalah alat sumber terbuka yang dapat Anda gunakan untuk berinteraksi dengan layanan AWS dari shell baris-perintah Anda.

2. Buat kebijakan yang dikelola pelanggan berikut yang memberikan izin untuk mengelola IDT dan peran AWS IoT Greengrass.

   Linux, macOS, or Unix

   ```
   aws iam create-policy --policy-name IDTUsageMetricsIAMPermissions --policy-
   document '{
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": [
                   "iot-device-tester:SendMetrics"
               ],
               "Resource": "*"
           }
       ]
   }'
   ```

Windows command prompt

```
aws iam create-policy --policy-name IDTUsageMetricsIAMPermissions --policy-
document
                                          '{\"Version\": \"2012-10-17\",
 \"Statement\": [{\"Effect\": \"Allow\", \"Action\": [\"iot-device-
tester:SendMetrics\"], \"Resource\": \"*\"}]}'
```

> **Note**
>
> Langkah ini mencakup contoh prompt perintah Windows karena menggunakan sintaks JSON yang berbeda dari perintah terminal Linux, MacOS, atau Unix.

3. Buat pengguna IAM dan lampirkan izin yang diperlukan oleh IDT untuk AWS IoT Greengrass.

   a. Buat pengguna IAM.

   ```
   aws iam create-user --user-name user-name
   ```

   b. Lampirkan kebijakan `IDTUsageMetricsIAMPermissions` yang Anda buat ke pengguna IAM Anda. Ganti *`user-name`* dengan nama pengguna IAM Anda dan *`<account-id>`* dalam perintah dengan ID Akun AWS Anda.

   ```
   aws iam attach-user-policy --user-name user-name --policy-arn
    arn:aws:iam::<account-id>:policy/IDTGreengrassIAMPermissions
   ```

4. Buat secret access key untuk pengguna tersebut.

   ```
   aws iam create-access-key --user-name user-name
   ```

   Simpan output tersebut di lokasi yang aman. Anda menggunakan informasi ini nanti untuk mengonfigurasi file kredensial AWS Anda.

Berikan kredensial AWS untuk IDT

Untuk membolehkan IDT mengakses kredensial AWS dan mengirimkan metrik ke AWS, lakukan hal berikut:

1. Simpan kredensial AWS untuk pengguna IAM Anda sebagai variabel lingkungan atau dalam file kredensial:

   a. Untuk menggunakan variabel lingkungan, jalankan perintah berikut:

   ```
   AWS_ACCESS_KEY_ID=access-key
   AWS_SECRET_ACCESS_KEY=secret-access-key
   ```

   b. Untuk menggunakan file kredensial, tambahkan informasi berikut ke `.aws/credentials file:`

   ```
   [profile-name]
   aws_access_key_id=access-key
   aws_secret_access_key=secret-access-key
   ```

2. Konfigurasikan bagian `auth` dari file `config.json`. Lihat informasi yang lebih lengkap di (Opsional) Konfigurasikan config.json.

IDT untuk AWS IoT Greengrass pemecahan masalah

IDT untuk AWS IoT Greengrass menulis eror ini ke berbagai lokasi berdasarkan jenis eror. Eror ditulis ke konsol tersebut, berkas log, dan laporan tes.

Kode eror

Tabel berikut mencantumkan kode eror yang dihasilkan oleh IDT untuk AWS IoT Greengrass.

Kode eror	Nama kode eror	Kemungkinan akar masalah	Pemecahan Masalah
101	InternalError	Terjadi eror internal.	Periksa log di bawah *<device-tester-extract-location>* `/results` direktori. Jika Anda tidak dapat men-debug masalah,

Kode eror	Nama kode eror	Kemungkinan akar masalah	Pemecahan Masalah
			hubungi Support DeveloperAWS.
102	TimeOutError	Tes tidak dapat diselesaikan dalam rentang waktu yang terbatas. Hal ini dapat terjadi jika: • Ada koneksi jaringan yang lambat antara mesin tes dan perangkat (sebagai contoh, jika Anda menggunakan jaringan VPN). • Jaringan yang lambat menunda komunikasi antara perangkat dan cloud. • Bidang `timeout` di file konfigurasi tes (`test.json`) telah diubah secara keliru.	• Periksa koneksi jaringan dan kecepatan. • Pastikan bahwa Anda tidak mengubah file apa pun di bawah `/test` direktori. • Coba jalankan grup tes gagal secara manual dengan `"--group-id"` bendera. • Coba jalankan tes suite dengan meningkatkan batas waktu tes. Untuk informasi selengkapnya, lihat Eror waktu habis.

Kode eror	Nama kode eror	Kemungkinan akar masalah	Pemecahan Masalah
103	PlatformNotSupportError	Kombinasi OS/arsitektur salah yang ditentukan dalam `device.json` .	Ubah konfigurasi Anda ke salah satu kombinasi yang didukung: • Linux, x86_64 • Linux, ARMv6l • Linux, ARMv7l • Linux, AArch64 • Ubuntu, x86_64 • OpenWRT, ARMv7l • OpenWRT, AArch64 Untuk informasi selengkapnya, lihat Konfigurasikan device.json.

Kode eror	Nama kode eror	Kemungkinan akar masalah	Pemecahan Masalah
104	VersionNotSupportError	Versi AWS IoT Greengrass perangkat lunak Core tidak didukung oleh versi IDT yang Anda gunakan.	Gunakan device_tester_bin version perintah untuk menemukan versi yang didukung AWS IoT Greengrass perangkat lunak Core. Sebagai contoh, jika Anda menggunakan macOS, gunakan ./devicetester_mac_x86_64 version. Untuk menemukan versi AWS IoT Greengrass perangkat lunak Inti yang Anda gunakan: • Jika anda menjalankan percobaan dengan prapasang AWS IoT Greengrass perangkat lunak Core, gunakan SSH untuk terhubung ke perangkat core AWS IoT Greengrass dan jalankan ***<path-to-preinstalled-green***

Kode eror	Nama kode eror	Kemungkinan akar masalah	Pemecahan Masalah
			grass-location> /greengrass/ggc/core/greengrassd --version • Jika Anda menjalankan tes dengan versi yang berbeda dari AWS IoT Greengrass perangkat lunak Core, pergi ke direktori `devicetester_greengrass_`*<os>*`/products/greengrass/gcc` ini. Vers AWS IoT Greengrass perangkat lunak Core adalah bagian dari nama file .zip. Anda dapat mengetes versi yang berbeda dari AWS IoT Greengrass perangkat lunak Core. Untuk informasi selengkapnya, lihat Mulai menggunakan AWS IoT Greengrass.

Kode eror	Nama kode eror	Kemungkinan akar masalah	Pemecahan Masalah
105	LanguageNotSupportError	IDT mendukung Python untuk AWS IoT Greengrass perpustakaan dan SDK saja.	Pastikan: • Paket SDK di bawah `devicetester_greengrass_ <os>/products/greengrass/ggsdk` adalah SDK Python. • Isi dari `bin` di bawah `devicetester_greengrass_ <os>/tests/GGQ_1.0.0/suite/resources/run.runtimefarm/bin` belum berubah.

Kode eror	Nama kode eror	Kemungkinan akar masalah	Pemecahan Masalah
106	ValidationError	Beberapa bidang di `device.json` atau `config.json` tidak valid.	Periksa pesan eror di sisi kanan kode eror dalam laporan. • Jenis autentikasi tidak valid untuk perangkat: Tentukan metode yang benar untuk terhubung ke perangkat Anda. Untuk informasi selengkapnya, lihat the section called “Konfigurasikan device.json”. • Jalur kunci pribadi tidak valid: Tentukan jalur yang benar ke kunci privat Anda. Untuk informasi selengkapnya, lihat Konfigurasikan device.json. • Tidak validWilayah AWS: Tentukan validWilayah AWSdalam`config.json` berkas. Untuk informasi selengkap

Kode eror	Nama kode eror	Kemungkinan akar masalah	Pemecahan Masalah
			nya, lihat AWS titik akhir layanan. • AWSkredensi: Atur validAWSkredensi pada mesin tes Anda (dengan menggunakan variabel lingkungan atau `credentials`). Verifikasi bahwa `auth` bidang dikonfigurasi dengan benar. Untuk informasi selengkapnya, lihat the section called “Buat dan konfigurasi pengguna Akun AWS”. • Input HSM tidak valid: Periksa `p11Provider` , `privateKeyLabel` , `slotLabel` , `slotUserPin` , dan `openSSLEngine` bidang di `device.json` .

Kode eror	Nama kode eror	Kemungkinan akar masalah	Pemecahan Masalah
107	SSHConnectionFailed	Mesin tes tidak dapat terhubung ke perangkat yang dikonfigurasi.	Verifikasi bahwa kolom berikut di file `device.json` Anda adalah benar: • `ip` • `user` • `privKeyPath` • `password` Untuk informasi selengkapnya, lihat Konfigurasikan device.json.

Kode eror	Nama kode eror	Kemungkinan akar masalah	Pemecahan Masalah
108	RunCommandError	Tes gagal untuk menjalankan perintah pada perangkat yang dites.	Verifikasi bahwa akses root diizinkan untuk pengguna terkonfigurasi di `device.json` . Kata sandi diperlukan oleh beberapa perangkat saat menjalankan perintah dengan akses root. Pastikan akses root diperbolehkan tanpa kata sandi. Untuk informasi lebih lanjut, lihat dokumentasi untuk perangkat Anda. Coba jalankan perintah gagal secara manual di perangkat Anda untuk melihat jika terjadi eror.
109	PermissionDeniedError	Tidak ada akses root.	Atur akses root untuk pengguna yang dikonfigurasi di perangkat Anda.
110	CreateFileError	Tidak dapat membuat file.	Periksa ruang disk dan izin direktori perangkat Anda.

Kode eror	Nama kode eror	Kemungkinan akar masalah	Pemecahan Masalah
111	CreateDirError	Tidak dapat membuat sebuah direktori.	Periksa ruang disk dan izin direktori perangkat Anda.
112	InvalidPathError	Jalur ke AWS IoT Greengrass perangkat lunak Core tidak benar.	Verifikasi bahwa jalur dalam pesan eror adalah valid. Jangan mengedit file apa pun di bawah direktori `devicetester_greengrass_` *<os>* ini.
113	InvalidFileError	Sebuah file tidak valid.	Verifikasi bahwa file dalam pesan eror adalah valid.

Kode eror	Nama kode eror	Kemungkinan akar masalah	Pemecahan Masalah
114	ReadFileError	File yang ditentukan tidak dapat dibaca.	Verifikasi hal berikut: • Izin file sudah benar. • `limits.config` mengizinkan file yang cukup untuk dibuka. • File yang ditentukan dalam pesan eror ada dan berlaku. Jika Anda mengetes di macOS, tingkatkan batas file yang terbuka. Batas default adalah 256, yang cukup untuk pengujian.

Kode eror	Nama kode eror	Kemungkinan akar masalah	Pemecahan Masalah
115	FileNotFoundError	File yang diperlukan tidak ditemukan.	Verifikasi hal berikut: • Sebuah file Greengrass terkompresi ada di bawah `devicetester_greengrass_ <os>/products/greengrass/ggc`. Anda dapat mengunduh file tar Core AWS IoT Greengrass dari halaman mengunduh AWS IoT Greengrass Perangkat Lunak Core ini. • Paket SDK ada di bawah `devicetester_greengrass_ <os>/products/greengrass/ggsdk`. • File di bawah `devicetester_greengrass_ <os>/`

Kode eror	Nama kode eror	Kemungkinan akar masalah	Pemecahan Masalah
			`tests` belum dimodifikasi.
116	OpenFileFailed	Tak dapat membuka file yang ditentukan.	Verifikasi hal berikut: • File yang ditentukan dalam pesan eror ada dan berlaku. • `limits.config` mengizinkan file yang cukup untuk dibuka. Jika Anda mengetes di macOS, tingkatkan batas file yang terbuka. Batas default adalah 256, yang cukup untuk pengujian.
117	WriteFileFailed	Gagal menulis file (dapat berupa DUT atau mesin tes).	Verifikasi bahwa direktori yang ditentukan dalam pesan eror yang ada dan bahwa Anda memiliki izin menulis.

Kode eror	Nama kode eror	Kemungkinan akar masalah	Pemecahan Masalah
118	FileCleanUpError	Sebuah tes gagal untuk menghapus file tertentu atau direktori atau untuk menjumlah file tertentu pada perangkat remote.	Jika file biner masih berjalan, file mungkin terkunci. Akhiri proses dan hapus file yang ditentukan.
119	InvalidInputError	Konfigurasi tidak valid.	Verifikasi bahwa `suite.json` file valid.
120	InvalidCredentialError	Kredensial AWS tidak valid.	• Verifikasi kredensial AWS Anda. Untuk informasi selengkapnya, lihat the section called “Konfigurasikan kredensial AWS Anda”. • Periksa koneksi jaringan Anda dan jalankan kembali grup tes. Masalah jaringan juga dapat casue eror ini.

Kode eror	Nama kode eror	Kemungkinan akar masalah	Pemecahan Masalah
121	AWSSessionError	Gagal membuat AWS sesi.	Eror ini dapat terjadi jika AWS kredensial tidak valid atau koneksi internet tidak stabil. Coba gunakan AWS CLI untuk memanggil AWS Operasi API.
122	AWSApicallError	Sebuah AWS eror API terjadi.	Eror ini mungkin terjadi karena masalah jaringan. Periksa jaringan Anda sebelum mencoba kembali grup tes.

Kode eror	Nama kode eror	Kemungkinan akar masalah	Pemecahan Masalah
123	IpNotExistError	Alamat IP tidak termasuk dalam informasi konektivitas.	Periksa koneksi internet Anda. Anda dapat menggunakan AWS IoT Greengrass konsol untuk memeriksa informasi konektivitas untuk AWS IoT Greengrass hal core yang sedang digunakan oleh tes. Jika ada 10 titik akhir yang disertakan dalam informasi konektivitas, Anda dapat menghapus beberapa atau semua dari mereka dan jalankan kembali tes. Untuk informasi selengkapnya, lihat informasi Konektivitas.
124	OTAJobNotCompleteError	Pekerjaan OTA tidak selesai.	Periksa koneksi internet Anda dan coba lagi grup tes OTA.

Kode eror	Nama kode eror	Kemungkinan akar masalah	Pemecahan Masalah
125	CreateGreengrassServiceRoleError	Salah satu dari hal berikut terjadi: • Terjadi eror saat membuat peran. • Terjadi eror saat melampirkan kebijakan untuk AWS IoT Greengrass peran layanan. • Kebijakan yang terkait dengan peran layanan tidak valid. • Terjadi eror ketika mengaitkan peran dengan Akun AWS.	Mengonfigurasi AWS IoT Greengrass peran layanan. Untuk informasi selengkapnya, lihat the section called “Peran layanan Greengrass”.
126	DependenciesNotPresentError	Satu atau lebih dependensi yang diperlukan untuk tes tertentu tidak ada pada perangkat.	Periksa log tes untuk melihat dependensi yang hilang di perangkat Anda: *<device-tester-extract-location>* /results/ *<execution-id>*/logs/*<test-case-name.log>*

Kode eror	Nama kode eror	Kemungkinan akar masalah	Pemecahan Masalah
127	InvalidHSMConfiguration	Konfigurasi HSM/PKCS yang disediakan tidak benar.	Di file `device.json` Anda, menyediakan konfigurasi yang diperlukan untuk berinteraksi dengan HSM menggunakan PKCS #11.

Kode eror	Nama kode eror	Kemungkinan akar masalah	Pemecahan Masalah

128	OTAJobNot SuccededError	Tugas OTA tidak berhasil.	• Jika Anda menjalankan grup tes `ota` secara individual, jalankan grup tes `ggcdependencies` untuk memverifikasi bahwa semua dependensi (seperti `wget`) hadir. Kemudian coba lagi `ota` grup tes.
			• Tinjau log terperinci di bawah *<device-tester-extract-location>* `/results/` *<execution-id>* `/logs/` untuk informasi pemecahan masalah dan eror. Khususnya, periksa log berikut ini:
			  • Log konsol (`test_manager.log`)
			  •

Kode eror	Nama kode eror	Kemungkinan akar masalah	Pemecahan Masalah
			Log tes kasus OTA (`ota_test.log`) • Log daemon GGC (`ota_test_ggc_logs.tar.gz`) • Log agen OTA (`ota_test_ota_logs.tar.gz`) • Periksa konektivitas internet Anda dan coba lagi `ota` grup tes. • Jika masalah berlanjut, hubungi Support Developer AWS.
129	NoConnectivityError	Agen host gagal untuk terhubung ke internet.	Periksa koneksi jaringan dan pengaturan firewall Anda. Coba lagi grup tes setelah masalah konektivitas teratasi.

Kode eror	Nama kode eror	Kemungkinan akar masalah	Pemecahan Masalah
130	NoPermissionError	User IAM yang Anda gunakan untuk menjalankan IDT AWS IoT Greengrass tidak memiliki izin untuk membuat AWS sumber daya yang diperlukan untuk menjalankan IDT.	Lihat Templat kebijakan izin untuk templat kebijakan yang memberikan izin yang diperlukan untuk menjalankan IDT AWS IoT Greengrass.

Kode eror	Nama kode eror	Kemungkinan akar masalah	Pemecahan Masalah
131	LeftoverAgentExistError	Perangkat Anda menjalankan AWS IoT Greengrass proses ketika Anda mencoba untuk mulai IDT AWS IoT Greengrass.	Pastikan tidak ada daemon Greengrass yang ada yang berjalan di perangkat Anda. • Anda dapat menggunakan perintah ini untuk menghentikan daemon: sudo ./*<absolute-path-to-greengrass-daemon>* / greengrassd stop. • Anda juga dapat mengakhiri daemon Greengrass dengan PID. **Note** Jika Anda menggunakan instalasi yang ada AWS IoT Greengrass dikonfigurasi untuk mulai secara otomatis

Kode eror	Nama kode eror	Kemungkinan akar masalah	Pemecahan Masalah
			setelah reboot, Anda harus menghentikan daemon setelah reboot dan sebelum menjalankan tes suite.
132	DeviceTimeOffsetError	Perangkat ini mempunyai waktu yang salah.	Atur perangkat Anda ke waktu yang benar.
133	InvalidMLConfiguration	Konfigurasi ML yang disediakan tidak benar.	Di file `device.json` Anda, sediakan konfigurasi yang benar yang diperlukan untuk menjalankan tes inferensi ML. Untuk informasi selengkapnya, lihat the section called “Opsional: Mengonfigurasi perangkat Anda untuk kualifikasi ML-nya”.

Menyelesaikan IDT untuk AWS IoT Greengrass eror

Ketika Anda menggunakan IDT, Anda harus mendapatkan file konfigurasi yang benar di tempat sebelum Anda menjalankan IDT untuk AWS IoT Greengrass. Jika Anda mendapatkan eror

penguraian dan konfigurasi, langkah pertama Anda adalah menemukan dan menggunakan templat konfigurasi yang sesuai untuk lingkungan Anda.

Jika Anda masih mengalami masalah, lihat proses debugging berikut.

Topik

- Di mana saya mencari eror?
- Penguraian eror
- Parameter yang diperlukan tidak ada eror
- Tidak dapat memulai eror pengetesan
- Tidak diotorisasi untuk mengakses eror sumber daya
- Izin ditolak eror
- Eror koneksi SSH
- Eror waktu habis
- Perintah tidak ditemukan eror saat pengetesan
- Pengecualian keamanan di macOS

Di mana saya mencari eror?

Eror tingkat tinggi ditampilkan di konsol selama pelaksanaan, dan ringkasan tes gagal dengan eror ditampilkan ketika semua tes selesai. `awsiotdevicetester_report.xml` berisi ringkasan dari semua eror yang menyebabkan tes gagal. Berkas log untuk setiap tes dilakukan akan disimpan dalam direktori yang bernama dengan UUID untuk pelaksanaan tes yang ditampilkan pada konsol tersebut selama tes dilakukan.

Direktori log tes terletak di *<device-tester-extract-location>*`/results/`*<execution-id>*`/logs/`. Direktori ini berisi file-file berikut, yang berguna untuk debugging.

File	Deskripsi
`test_manager.log`	Semua log yang ditulis ke konsol tersebut selama pelaksanaan tes. Ringkasan hasil terletak di akhir file ini, yang mencakup daftar tes yang gagal.

File	Deskripsi
	Log peringatan dan eror dalam file ini dapat memberikan beberapa informasi tentang kegagalan.
<test-group-id> __*<test-name>* .log	Log terperinci untuk tes tertentu.
<test-name> _ggc_logs.tar.gz	Koleksi terkompresi dari semua log AWS IoT Greengrass core daemon yang dihasilkan selama pengetesan. Untuk informasi selengkapnya, lihat Pemecahan Masalah AWS IoT Greengrass.
<test-name> _ota_logs.tar.gz	Koleksi terkompresi log yang dihasilkan oleh AWS IoT Greengrass agen OTA selama tes. Untuk tes OTA saja.
<test-name> _basic_assertion_publisher_ggad_logs.tar.gz	Koleksi terkompresi log yang dihasilkan oleh AWS IoT perangkat penerbit selama pengetesan.
<test-name> _basic_assertion_subscriber_ggad_logs.tar.gz	Koleksi terkompresi log yang dihasilkan oleh AWS IoT perangkat pelanggan selama pengetesan.

Penguraian eror

Kadang-kadang, eror ketik dalam konfigurasi JSON dapat menyebabkan penguraian eror. Sebagian besar waktu, masalah adalah hasil dari menghilangkan braket, koma, atau tanda kutip dari file JSON Anda. IDT melakukan validasi JSON dan mencetak informasi debugging. Ini mencetak garis di mana eror terjadi, nomor baris, dan nomor kolom eror sintaks. Informasi ini harus cukup untuk membantu Anda memperbaiki eror, tetapi jika Anda masih tidak dapat menemukan eror, Anda dapat melakukan validasi secara manual di IDE Anda, editor teks seperti Atom atau Sublime, atau melalui alat online seperti JsonLint.

Parameter yang diperlukan tidak ada eror

Karena fitur baru sedang ditambahkan ke IDT, perubahan pada file konfigurasi mungkin akan diperkenalkan. Menggunakan file konfigurasi lama mungkin merusak konfigurasi Anda. Jika hal ini terjadi, *<test_case_id>*`.log` file di bawah `/results/`*<execution-id>*`/logs` secara eksplisit mencantumkan semua parameter yang hilang. IDT juga memvalidasi skema file konfigurasi JSON Anda untuk memastikan bahwa versi terbaru yang didukung telah digunakan.

Tidak dapat memulai eror pengetesan

Anda mungkin mengalami eror yang mengarah ke kegagalan selama tes dimulai. Ada beberapa kemungkinan penyebabnya, jadi lakukan hal berikut:

- Pastikan bahwa nama kolam Anda termasuk dalam perintah pelaksanaan Anda benar-benar ada. Nama kolam direferensikan langsung dari `device.json` file.
- Pastikan bahwa perangkat di kolam renang Anda memiliki parameter konfigurasi yang benar.

Tidak diotorisasi untuk mengakses eror sumber daya

Anda mungkin melihat `<user or role> is not authorized to access this resource` pesan eror dalam output terminal atau di `test_manager.log` file di bawah `/results/`*<execution-id>*`/logs`. Untuk mengatasi masalah ini, lampirkan `AWSIoTDeviceTesterForGreengrassFullAccess` kebijakan terkelola untuk pengguna pengetesan Anda. Untuk informasi selengkapnya, lihat the section called “Buat dan konfigurasi pengguna Akun AWS”.

Izin ditolak eror

IDT melakukan operasi pada berbagai direktori dan file dalam perangkat yang diuji. Beberapa operasi ini memerlukan akses akar. Untuk mengotomatisasi operasi ini, IDT harus dapat menjalankan perintah dengan sudo tanpa memasukkan kata sandi.

Ikuti langkah-langkah ini untuk mengizinkan akses sudo tanpa mengetikkan kata sandi.

> **Note**
>
> `user` dan `username` mengacu pada pengguna SSH yang digunakan oleh IDT untuk mengakses perangkat yang diuji.

1. Gunakan sudo usermod -aG sudo ***<ssh-username>*** untuk menambahkan pengguna SSH Anda ke grup sudo.
2. Keluar, lalu masuk agar perubahan diterapkan.
3. Buka file `/etc/sudoers` dan tambahkan baris berikut ke akhir file: *<ssh-username>* `ALL=(ALL) NOPASSWD: ALL`

> **Note**
>
> Sebagai praktik terbaik, kami merekomendasikan Anda menggunakan sudo visudo ketika Anda mengedit `/etc/sudoers`.

Eror koneksi SSH

Ketika IDT tidak dapat terhubung ke perangkat yang dites, kegagalan koneksi dicatat di `/results/`*<execution-id>*`/logs/`*<test-case-id>*`.log`. Pesan kegagalan SSH muncul di bagian atas berkas log ini karena menghubungkan ke perangkat yang sedang dites adalah salah satu operasi pertama yang dilakukan IDT.

Sebagian besar pengaturan Windows menggunakan aplikasi terminal Putty untuk terhubung ke host Linux. Aplikasi ini mengharuskan file kunci privat PEM standar diubah menjadi format Windows berpemilik yang disebut PPK. Ketika IDT dikonfigurasi di file `device.json` Anda, gunakan file PEM saja. Jika Anda menggunakan file PPK, IDT tidak dapat membuat koneksi SSH dengan AWS IoT Greengrass dan tidak dapat menjalankan tes run.

Eror waktu habis

Anda dapat meningkatkan waktu habis untuk setiap tes dengan menentukan pengganda waktu habis, yang diterapkan ke nilai default dari waktu habis setiap tes. Nilai apa pun yang dikonfigurasi untuk bendera ini harus lebih besar dari atau sama dengan 1.0.

Untuk menggunakan pengganda batas waktu, gunakan bendera `--timeout-multiplier` saat menjalankan tes. Misalnya:

```
./devicetester_linux run-suite --suite-id GGQ_1.0.0 --pool-id DevicePool1 --timeout-
multiplier 2.5
```

Untuk informasi lebih lanjut, jalankan `run-suite --help`.

Perintah tidak ditemukan eror saat pengetesan

Anda memerlukan versi lama dari pustaka OpenSSL (libssl1.0.0) untuk menjalankan pengujian pada perangkat AWS IoT Greengrass. Sebagian besar distribusi Linux menggunakan libssl versi 1.0.2 atau yang lebih baru (v1.1.0).

Sebagai contoh, pada Raspberry Pi, jalankan perintah berikut untuk menginstal versi libssl yang diperlukan:

1. ```
   wget http://ftp.us.debian.org/debian/pool/main/o/openssl/
   libssl1.0.0_1.0.2l-1~bpo8+1_armhf.deb
   ```
2. ```
   sudo dpkg -i libssl1.0.0_1.0.2l-1~bpo8+1_armhf.deb
   ```

Pengecualian keamanan di macOS

Saat Anda menjalankan IDT di mesin host yang menggunakan macOS 10.15, tiket notaris untuk IDT tidak terdeteksi dengan benar dan IDT diblokir agar tidak dijalankan. Untuk menjalankan IDT, Anda akan perlu untuk memberikan pengecualian keamanan untuk `devicetester_mac_x86-64` dilaksanakan.

Untuk memberikan pengecualian keamanan untuk IDT executable

1. Luncurkan Sistem Preferensi dari menu Apple.
2. Pilih Keamanan & Privasi, kemudian pada tab Umum, klik ikon kunci untuk membuat perubahan pada pengaturan keamanan.
3. Cari pesan `"devicetester_mac_x86-64" was blocked from use because it is not from an identified developer.` dan pilih Mengizinkan Bagaimanapun.
4. Menerima peringatan keamanan.

Jika Anda memiliki pertanyaan tentang kebijakan dukungan IDT, hubungi AWS Dukungan Pelanggan.

Support kebijakan untuk AWS IoT Penguji Perangkat untuk AWS IoT Greengrass V1

AWS IoTDevice Tester (IDT) untukAWS IoT Greengrassadalah kerangka pengujian yang dapat diunduh yang memungkinkan Anda untuk memvalidasi danmemenuhi syarat-muAWS IoT Greengrassperangkat untuk dimasukkan ke dalamAWS PartnerKatalog Perangkat. Kami menyarankan Anda menggunakan versi terbaru dariAWS IoT Greengrassdan IDT untuk menguji atau memenuhi syarat perangkat Anda. Untuk informasi selengkapnya, lihatVersi IDT yang didukung untukAWS IoT Greengrass V2diAWS IoT Greengrass Version 2Panduan Pengembang.

Anda juga dapat menggunakan salah satu versiAWS IoT Greengrassdan IDT untuk menguji atau memenuhi syarat perangkat Anda. Meskipun Anda dapat terus menggunakanversi IDT yang tidak didukung, versi tersebut tidak menerima perbaikan bug atau pembaruan.

> **Important**
>
> Pada 4 April 2022,AWS IoTDevice Tester (IDT) untukAWS IoT Greengrass V1tidak lagi menghasilkan laporan kualifikasi yang ditandatangani. Anda tidak dapat lagi memenuhi syarat baruAWS IoT Greengrass V1perangkat untuk daftar diAWS PartnerKatalog Perangkatmelaluiprogram Kualifikasi Perangkat. Meskipun Anda tidak dapat memenuhi syarat perangkat Greengrass V1, Anda dapat terus menggunakan IDT untukAWS IoT Greengrass V1untuk menguji perangkat Greengrass V1 Anda. Kami menyarankan agar Anda menggunakanIDT untukAWS IoT Greengrass V2untuk memenuhi syarat dan daftar perangkat Greengrass diAWS PartnerKatalog Perangkat.

Jika Anda memiliki pertanyaan tentang kebijakan support, kontak AWS Dukungan Pelanggan.

Pemecahan Masalah AWS IoT Greengrass

Bagian ini menyediakan informasi pemecahan masalah dan solusi yang mungkin untuk membantu menyelesaikan masalah dengan AWS IoT Greengrass.

Untuk informasi tentang AWS IoT Greengrass kuota (batas), lihat Service Quotas di. Referensi Umum Amazon Web Services

AWS IoT Greengrass Masalah core

Jika perangkat lunak Core AWS IoT Greengrass tidak memulai, cobalah langkah-langkah pemecahan masalah umum berikut ini:

- Pastikan bahwa Anda menginstal binari yang sesuai untuk arsitektur Anda. Untuk informasi selengkapnya, lihat AWS IoT Greengrass Perangkat lunak Core.
- Pastikan perangkat core Anda memiliki penyimpanan lokal yang tersedia. Untuk informasi selengkapnya, lihat the section called “Pemecahan masalah penyimpanan”.
- Periksa `runtime.log` dan `crash.log` untuk pesan error. Untuk informasi selengkapnya, lihat the section called “Pemecahan masalah dengan catatan”.

Cari gejala dan error berikut untuk menemukan informasi untuk membantu memecahkan masalah dengan AWS IoT Greengrass core.

Masalah

- Kesalahan: File konfigurasi hilang CaPath, CertPath atau KeyPath. <pid>Proses daemon Greengrass dengan [pid =] mati.
- Error: Gagal mengurai/<greengrass-root> /config/config.json.
- Kesalahan: Terjadi kesalahan saat membuat konfigurasi TLS: UrisCheme ErrUnknown
- Error: Waktu aktif gagal untuk mulai: tidak dapat mulai pekerja: pengujian kontainer habis waktu.
- <address>Kesalahan: Gagal memanggil PutLogEvents di Cloudwatch lokal, LogGroup:/GreengrassSystem/connection_manager, kesalahan:: permintaan kirim gagal disebabkan oleh: Posting http RequestError://<path>/cloudwatch/logs/: dial tcp: getsockopt: koneksi ditolak, respons: {}.

- <region><account-id><function-name><version><file-name>Kesalahan: Tidak dapat membuat server karena: gagal memuat grup: chmod/<greengrass-root>/ggc/deployment/lambda/ arn:aws:lambda: ::function::/: tidak ada file atau direktori seperti itu.
- Perangkat lunak Core AWS IoT Greengrass tidak mulai setelah Anda berubah dari berjalan tanpa kontainerisasi untuk berjalan dalam kontainer Greengrass.
- error: Ukuran spool harus setidaknya 262144 byte.
- Kesalahan: [KESALAHAN]-kesalahan olahpesan cloud: Terjadi kesalahan ketika mencoba menerbitkan pesan. {"errorString": "timed out operasi"}
- <version>Kesalahan: container_linux.go: 344: memulai proses kontainer menyebabkan “process_linux.go: 424: container init disebabkan\" rootfs_linux.go:64: mounting\\\” /greengrass/ggc/ socket/greengrass_ipc.sock\\\” ke rootfs\\\” /greengrass/ggc/packageses/ /rootfs/merged\\\” di\\\” / greengrass_ipc.sock\\\” disebabkan\\\ "stat /greengrass/ggc/socket/greengrass_ipc.sock: izin ditolak \\\"\ "”.
- error: Greengrass daemon berjalan dengan PID: <process-id>. Beberapa komponen sistem gagal untuk mulai. Periksa 'runtime.catatan' untuk error.
- Bayangan perangkat tidak sinkron dengan cloud.
- ERROR: tidak dapat menerima koneksi TCP. menerima tcp [::]:8000 accept4: terlalu banyak file terbuka.
- Error: error eksekusi waktu aktif: tidak dapat memulai kontainer lambda. container_linux.go: 259: memulai proses kontainer disebabkan "process_linux.go:345: container init disebabkan \"rootfs_linux.go:50: mempersiapkan rootfs disebabkan \\\ "izin ditolak\\\"\"".
- Peringatan: [PERINGATAN] - [5] GK Remote: Kesalahan mengambil data kunci publik: ErrPrincipalNotConfigured: kunci pribadi untuk tidak MqttCertificate disetel.
- <account-id><role-name><region>Kesalahan: Izin ditolak saat mencoba menggunakan peran arn:aws:iam: ::role/ untuk mengakses url s3 https://-greengrass-updates.s3. <region><architecture><distribution-version>.amazonaws.com/core/ /greengrass-core- .tar.gz.
- Core AWS IoT Greengrass dikonfigurasikan untuk menggunakan proksi jaringan dan fungsi Lambda Anda tidak dapat membuat koneksi keluar.
- Core adalah dalam loop connect-disconnect yang tak terbatas. File runtime.log berisi serangkaian kontinyu entri menghubungkan dan memutuskan.
- Error: tidak dapat memulai kontainer lambda. container_linux.go:259: memulai proses kontainer disebabkan "process_linux.go:345: container init disebabkan \"rootfs_linux.go: 62: mounting\\\ "proc \\\" to rootfs\\\"

- [ERROR]-error ekseskusi waktu aktif: tidak dapat memulai kontainer lambda. {"errorString": "gagal menginisialisasi pemasangan kontainer: gagal untuk menutupi root greengrass di overlay dir atas: gagal membuat perangkat mask pada direktori <ggc-path>: file ada"}
- [ERROR] -Penerapan gagal. {"deploymentID": <deployment-id>"", "errorString": "proses pengujian kontainer dengan <pid>pid gagal: status proses kontainer: status keluar 1"}
- Error: [ERROR]-error eksekusi waktu aktif: tidak dapat memulai kontainer lambda. {"ErrorString": "gagal menginisialisasi pemasangan kontainer: gagal membuat overlay fs untuk kontainer: pemasangan overlay pada /greengrass/ggc/packages/ <ggc-version> /rootfs/digabung gagal: gagal untuk me-mount dengan sumber args=\"no_source\" dest=\"/greengrass/ggc/packages/<ggc-version>/rootfs/merged\" fstype=\"overlay\" flags=\"0\" data=\"lowerdir=/greengrass/ggc/packages/<ggc-version>/dns:/,upperdir=/greengrass/ggc/packages/<ggc-version>/rootfs/upper,workdir=/greengrass/ggc/packages/<ggc-version>/rootfs/work\": terlalu banyak tingkatan tautan simbolik"}
- Error: [DEBUG]-Gagal untuk mendapatkan rute. Membuang pesan.
- Error: [Errno 24] Terlalu banyak membuka <lambda-function>, [Errno 24] Terlalu banyak membuka file
- Kesalahan: server ds gagal mulai mendengarkan soket: dengarkan unix <ggc-path>/ggc/socket/greengrass_ipc.sock: bind: argumen tidak valid
- [INFO] (Mesin Fotokopi) aws.greengrass. StreamManager: stdout. Disebabkan oleh: com.fasterxml.jackson.databind. JsonMappingException: Instan melebihi minimum atau maksimum instan
- GPG error: https://dnw9lb6lzp2d8.cloudfront.net stable InRelease: The following signatures were invalid: EXPKEYSIG 68D644ABD2327D47 AWS Greengrass Master Key

Kesalahan: File konfigurasi hilang CaPath, CertPath atau KeyPath. <pid>Proses daemon Greengrass dengan [pid =] mati.

Solusi: Anda mungkin melihat kesalahan ini di `crash.log` ketika AWS IoT Greengrass perangkat lunak core tidak mulai. Hal ini dapat terjadi jika Anda menjalankan v1.6 atau sebelumnya. Lakukan salah satu dari cara berikut:

- Upgade ke v1.7 atau lebih baru. Kami merekomendasikan bahwa Anda selalu menjalankan versi terbaru AWS IoT Greengrass perangkat lunak Core. Untuk informasi unduhan, lihat AWS IoT Greengrass Perangkat lunak Core.

- Gunakan format `config.json` yang benar untuk AWS IoT Greengrass rersi perangkat lunak Core. Untuk informasi selengkapnya, lihat the section called “AWS IoT Greengrass file konfigurasi core”.

> **Note**
>
> Untuk menemukan versi AWS IoT Greengrass perangkat lunak core diinstal pada perangkat core, jalankan perintah berikut di terminal perangkat Anda.
>
> ```
> cd /greengrass-root/ggc/core/
> sudo ./greengrassd --version
> ```

Error: Gagal mengurai/<greengrass-root> /config/config.json.

Solusi: Anda mungkin melihat kesalahan ini ketika perangkat lunak Core AWS IoT Greengrass tidak mulai. Pastikan File konfigurasi Greengrass menggunakan format JSON yang valid.

Buka `config.json` (terletak di */greengrass-root*`/config`) dan memvalidasi format JSON. Sebagai contoh, pastikan bahwa koma digunakan dengan benar.

Kesalahan: Terjadi kesalahan saat membuat konfigurasi TLS: UrisCheme ErrUnknown

Solusi: Anda mungkin melihat kesalahan ini ketika perangkat lunak Core AWS IoT Greengrass tidak mulai. Pastikan properti di Kripto bagian dari file konfigurasi Greengrass valid. Pesan error harus menyediakan informasi lebih lanjut.

Buka `config.json` (terletak di */greengrass-root*`/config`) dan periksa `crypto` bagian. Sebagai contoh, sertifikat dan jalur kunci harus menggunakan format URI yang benar dan arahkan ke lokasi yang benar.

Error: Waktu aktif gagal untuk mulai: tidak dapat mulai pekerja: pengujian kontainer habis waktu.

Solusi: Anda mungkin melihat kesalahan ini ketika perangkat lunak Core AWS IoT Greengrass tidak mulai. Atur `postStartHealthCheckTimeout` properti di file konfigurasi Greengrass. Properti opsional ini mengonfigurasi jumlah waktu (dalam milidetik) yang Greengrass daemon menunggu pemeriksaan kondisi pasca-mulai untuk menyelesaikan. Nilai default adalah 30 detik (30000 ms).

Buka `config.json` (terletak di */greengrass-root*`/config`). Di `runtime` objek, tambahkan `postStartHealthCheckTimeout` properti dan atur nilai ke nomor yang lebih besar dari 30000. Tambahkan koma di mana diperlukan untuk membuat dokumen JSON valid. Sebagai contoh:

```
...
"runtime" : {
  "cgroup" : {
    "useSystemd" : "yes"
  },
  "postStartHealthCheckTimeout" : 40000
},
...
```

<address>Kesalahan: Gagal memanggil PutLogEvents di Cloudwatch lokal, LogGroup:/GreengrassSystem/connection_manager, kesalahan:: permintaan kirim gagal disebabkan oleh: Posting http RequestError:// <path>/cloudwatch/logs/: dial tcp: getsockopt: koneksi ditolak, respons: {}.

Solusi: Anda mungkin melihat kesalahan ini ketika perangkat lunak Core AWS IoT Greengrass tidak mulai. Hal ini dapat terjadi jika Anda menjalankan AWS IoT Greengrass pada Raspberry Pi dan setup memori yang diperlukan belum selesai. Untuk informasi selengkapnya, lihat langkah ini.

<region><account-id><function-name><version><file-name>Kesalahan: Tidak dapat membuat server karena: gagal memuat grup: chmod/<greengrass-root>/ggc/deployment/lambda/arn:aws:lambda: ::function::/: tidak ada file atau direktori seperti itu.

Solusi: Anda mungkin melihat kesalahan ini ketika perangkat lunak Core AWS IoT Greengrass tidak mulai. Jika Anda men-deploy executable Lambda ke core, periksa fungsi properti `Handler` di file `group.json` ini (terletak di /*greengrass-root*/ggc/deployment/grup). Jika handler bukan nama yang tepat dari compiled executable Anda, ganti konten `group.json` file dengan obyek JSON kosong (`{}`), dan jalankan perintah berikut untuk mulai AWS IoT Greengrass:

```
cd /greengrass/ggc/core/
sudo ./greengrassd start
```

Kemudian, gunakan AWS Lambda API untuk memperbarui fungsi parameter `handler` konfigurasi, menerbitkan versi fungsi baru, dan update alias. Untuk informasi lebih lanjut, lihat AWS Lambda fungsi versioning dan aliases.

Dengan asumsi bahwa Anda menambahkan fungsi ke grup Greengrass Anda dengan alias (disarankan), Anda sekarang dapat men-deploy ulang grup Anda. (Jika tidak, Anda harus menunjuk ke versi fungsi baru atau alias dalam definisi grup dan langganan sebelum Anda men-deploy grup.)

Perangkat lunak Core AWS IoT Greengrass tidak mulai setelah Anda berubah dari berjalan tanpa kontainerisasi untuk berjalan dalam kontainer Greengrass.

Solusi: Periksa bahwa Anda tidak kehilangan kontainer dependensi apa pun.

error: Ukuran spool harus setidaknya 262144 byte.

Solusi: Anda mungkin melihat kesalahan ini ketika perangkat lunak Core AWS IoT Greengrass tidak mulai. Buka `group.json` file (terletak di */greengrass-root*/ggc/deployment/group), ganti

konten file dengan objek JSON kosong ({}), dan jalankan perintah berikut untuk meluncurkan AWS IoT Greengrass:

```
cd /greengrass/ggc/core/
sudo ./greengrassd start
```

Kemudian ikuti langkah-langkah di the section called “Untuk cache pesan di penyimpanan lokal” prosedur. Untuk `GGCloudSpooler` fungsi, pastikan untuk menentukan `GG_CONFIG_MAX_SIZE_BYTES` nilai yang lebih dari atau sama dengan 262144.

Kesalahan: [KESALAHAN]-kesalahan olahpesan cloud: Terjadi kesalahan ketika mencoba menerbitkan pesan. {"errorString": "timed out operasi"}

Solusi: Anda mungkin melihat kesalahan ini di `GGCloudSpooler.log` ketika Greengrass core tidak dapat mengirim pesan MQTT ke AWS IoT Core. Hal ini dapat terjadi jika lingkungan core memiliki bandwidth terbatas dan latency tinggi. Jika Anda menjalankan AWS IoT Greengrass v1.10.2 atau yang lebih baru, coba tingkatkan nilai `mqttOperationTimeout` di file config.json ini. Jika properti tidak hadir, tambahkan nilai ke `coreThing` Objek. Sebagai contoh:

```
{
  "coreThing": {
    "mqttOperationTimeout": 10,
    "caPath": "root-ca.pem",
    "certPath": "hash.cert.pem",
    "keyPath": "hash.private.key",
    ...
  },
  ...
}
```

Nilai default adalah 5 dan nilai minimum adalah 5.

<version>Kesalahan: container_linux.go: 344: memulai proses kontainer menyebabkan “process_linux.go: 424: container init disebabkan\" rootfs_linux.go:64: mounting\\\” /greengrass/ggc/socket/greengrass_ipc.sock \\\” ke rootfs\\\” /greengrass/ggc/packageses/ /rootfs/merged\\\” di\\\” / greengrass_ipc.sock\\\” disebabkan\\\ "stat /greengrass/ggc/socket/ greengrass_ipc.sock: izin ditolak\\\"\ "”.

Solusi: Anda mungkin melihat kesalahan ini di `runtime.log` ketika AWS IoT Greengrass perangkat lunak Core tidak mulai. Hal ini terjadi jika `umask` lebih tinggi dari `0022`. Untuk mengatasi masalah ini, Anda harus mengatur `umask` ke `0022` atau lebih rendah. Nilai `0022` memberikan semua orang izin baca ke file baru secara default.

error: Greengrass daemon berjalan dengan PID: <process-id>. Beberapa komponen sistem gagal untuk mulai. Periksa 'runtime.catatan' untuk error.

Solusi: Anda mungkin melihat kesalahan ini ketika perangkat lunak Core AWS IoT Greengrass tidak mulai. Periksa `runtime.log` dan `crash.log` untuk informasi error spesifik. Untuk informasi selengkapnya, lihat the section called “Pemecahan masalah dengan catatan”.

Bayangan perangkat tidak sinkron dengan cloud.

Solusi: Pastikan bahwa AWS IoT Greengrass memiliki izin untuk tindakan `iot:UpdateThingShadow` dan `iot:GetThingShadow` di peran layanan Greengrass. Jika peran layanan menggunakan `AWSGreengrassResourceAccessRolePolicy` kebijakan terkelola, izin ini disertakan secara default.

Lihat Memecahkan masalah timeout sinkronisasi bayangan.

ERROR: tidak dapat menerima koneksi TCP. menerima tcp [::]:8000 accept4: terlalu banyak file terbuka.

Solusi: Anda mungkin melihat kesalahan ini di `greengrassd` output skrip. Hal ini dapat terjadi jika batas file descriptor untuk AWS IoT Greengrass perangkat lunak core telah mencapai ambang batas dan harus ditingkatkan.

Gunakan perintah berikut ini dan kemudian restart AWS IoT Greengrass perangkat lunak Core.

```
ulimit -n 2048
```

> **Note**
>
> Dalam contoh ini, batas meningkat menjadi 2048. Pilih nilai yang sesuai untuk kasus penggunaan Anda.

Error: error eksekusi waktu aktif: tidak dapat memulai kontainer lambda. container_linux.go: 259: memulai proses kontainer disebabkan "process_linux.go:345: container init disebabkan\"rootfs_linux.go:50: mempersiapkan rootfs disebabkan \\\ "izin ditolak\\\"\"".

Solusi: Baik instal AWS IoT Greengrass langsung di bawah direktori root, atau pastikan bahwa direktori di mana AWS IoT Greengrass perangkat lunak core diinstal dan direktori induknya memiliki `execute` izin untuk semua orang.

Peringatan: [PERINGATAN] - [5] GK Remote: Kesalahan mengambil data kunci publik: ErrPrincipalNotConfigured: kunci pribadi untuk tidak MqttCertificate disetel.

Solusi: AWS IoT Greengrass gunakan handler umum untuk memvalidasi properti dari semua prinsip keamanan. Peringatan ini di `runtime.log` diharapkan kecuali Anda menentukan kunci pribadi

khusus untuk server MQTT lokal. Untuk informasi selengkapnya, lihat the section called “Keamanan utama”.

<account-id><role-name><region>Kesalahan: Izin ditolak saat mencoba menggunakan peran arn:aws:iam: ::role/ untuk mengakses url s3 https://-greengrass-updates.s3. <region><architecture><distribution-version>.amazonaws.com/core/ /greengrass-core- .tar.gz.

Solusi: Anda mungkin melihat kesalahan ini ketika pembaruan over-the-air (OTA) gagal. Dalam kebijakan peran signer, tambahkan target Wilayah AWS sebagai `Resource`. Peran signer ini digunakan untuk presign S3 URL untuk AWS IoT Greengrass memperbarui perangkat lunak. Untuk informasi selengkapnya, lihat. Peran signer URL S3.

Core AWS IoT Greengrass dikonfigurasikan untuk menggunakan proksi jaringan dan fungsi Lambda Anda tidak dapat membuat koneksi keluar.

Solusi: Tergantung pada waktu aktif dan executable yang digunakan oleh fungsi Lambda untuk membuat koneksi, Anda mungkin juga menerima error timeout koneksi. Pastikan fungsi Lambda anda menggunakan konfigurasi proksi yang sesuai untuk menyambung melalui proksi rangkaian. AWS IoT Greengrass melewati konfigurasi proksi ke fungsi Lambda yang ditetapkan pengguna melalui `http_proxy`, `https_proxy`, dan `no_proxy` variabel lingkungan. Mereka dapat diakses seperti yang ditunjukkan dalam potongan Python berikut.

```
import os
print(os.environ['http_proxy'])
```

Gunakan kasus yang sama seperti variabel didefinisikan dalam lingkungan Anda, misalnya, semua huruf kecil `http_proxy` atau semua huruf besar `HTTP_PROXY`. Untuk variabel ini, AWS IoT Greengrass mendukung keduanya.

> **Note**
>
> Perpustakaan yang paling umum digunakan untuk membuat koneksi (seperti boto3 atau cURL dan paket `requests` python) menggunakan variabel lingkungan ini secara default.

Core adalah dalam loop connect-disconnect yang tak terbatas. File runtime.log berisi serangkaian kontinyu entri menghubungkan dan memutuskan.

Solusi: Hal ini dapat terjadi ketika perangkat lain hard-coded untuk menggunakan nama hal core sebagai ID klien untuk koneksi MQTT ke AWS IoT. Sambungan serentak yang sama Wilayah AWS dan Akun AWS harus menggunakan ID klien yang unik. Secara default, core menggunakan nama hal core yang sama sebagai ID klien untuk koneksi ini.

Untuk mengatasi masalah ini, Anda dapat mengubah ID klien yang digunakan oleh perangkat lain untuk sambungan (disarankan) atau menimpa nilai default untuk core.

Untuk menimpa ID klien default untuk perangkat core

1. Jalankan perintah berikut untuk menghentikan Greengrass daemon:

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

2. Buka *greengrass-root*`/config/config.json` untuk diedit sebagai pengguna su.
3. Di `coreThing` objek, tambahkan `coreClientId` properti, dan atur nilai untuk ID klien khusus Anda. Nilainya harus antara 1 sampai 128 karakter. Itu harus unik dalam arus Wilayah AWS untuk Akun AWS.

   ```
   "coreClientId": "MyCustomClientId"
   ```

4. Mulai daemon.

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

Error: tidak dapat memulai kontainer lambda. container_linux.go:259: memulai proses kontainer disebabkan "process_linux.go:345: container init disebabkan \"rootfs_linux.go: 62: mounting\\\ "proc\\\" to rootfs\\\"

Solusi: Pada beberapa platform, Anda mungkin melihat kesalahan ini di `runtime.log` ketika AWS IoT Greengrass mencoba untuk me-mount `/proc` sistem file untuk membuat kontainer Lambda. Atau, Anda mungkin melihat kesalahan serupa, seperti `operation not permitted` atau `EPERM`. Error ini dapat terjadi bahkan jika pengujian berjalan pada platform oleh dependensi checker skrip lulus.

Cobalah salah satu solusi berikut:

- Aktifkan `CONFIG_DEVPTS_MULTIPLE_INSTANCES` opsi di kernel Linux.
- Set `/proc` opsi mount pada host untuk `rw,relatim` saja.
- Upgrade kernel Linux menjadi 4.9 atau yang lebih baru.

> **Note**
>
> Masalah ini tidak terkait dengan pemasangan `/proc` untuk akses sumber daya lokal.

[ERROR]-error ekseskusi waktu aktif: tidak dapat memulai kontainer lambda. {"errorString": "gagal menginisialisasi pemasangan kontainer: gagal untuk menutupi root greengrass di overlay dir atas: gagal membuat perangkat mask pada direktori <ggc-path>: file ada"}

Solusi: Anda mungkin melihat kesalahan ini di runtime.catatan saat deployment gagal. Error ini terjadi jika fungsi Lambda di grup AWS IoT Greengrass tidak dapat mengakses direktori `/usr` dalam sistem file core.

Untuk mengatasi masalah ini, tambahkan sumber daya volume lokal ke grup dan kemudian men-deploy grup. Sumber daya ini harus:

- Tentukan `/usr` sebagai Jalur sumber dan Jalur tujuan.
- Secara otomatis menambahkan izin grup OS dari grup Linux yang memiliki sumber daya.
- Berafiliasi dengan fungsi Lambda dan mengizinkan akses hanya baca.

[ERROR] -Penerapan gagal. {"deploymentID”: <deployment-id>"“, “errorString”: “proses pengujian kontainer dengan <pid>pid gagal: status proses kontainer: status keluar 1"}

Solusi: Anda mungkin melihat kesalahan ini di runtime.catatan saat deployment gagal. Error ini terjadi jika fungsi Lambda di grup AWS IoT Greengrass tidak dapat mengakses direktori `/usr` dalam sistem file core.

Anda dapat mengkonfirmasi bahwa ini adalah kasus dengan memeriksa `GGCanary.log` untuk error tambahan. Jika fungsi Lambda tidak dapat mengakses `/usr` direktori, `GGCanary.log` akan berisi error berikut:

```
[ERROR]-standard_init_linux.go:207: exec user process caused "no such file or
 directory"
```

Untuk mengatasi masalah ini, tambahkan sumber daya volume lokal ke grup dan kemudian men-deploy grup. Sumber daya ini harus:

- Tentukan `/usr` sebagai Jalur sumber dan Jalur tujuan.
- Secara otomatis menambahkan izin grup OS dari grup Linux yang memiliki sumber daya.
- Berafiliasi dengan fungsi Lambda dan mengizinkan akses hanya baca.

Error: [ERROR]-error eksekusi waktu aktif: tidak dapat memulai kontainer lambda. {"ErrorString": "gagal menginisialisasi pemasangan kontainer: gagal membuat overlay fs untuk kontainer: pemasangan overlay pada / greengrass/ggc/packages/ <ggc-version> /rootfs/digabung gagal: gagal untuk me-mount dengan sumber args=\"no_source\" dest=\"/greengrass/ggc/packages/<ggc-version>/rootfs/merged\" fstype=\"overlay\" flags=\"0\" data=\"lowerdir=/greengrass/ggc/packages/<ggc-version>/dns:/,upperdir=/greengrass/ggc/packages/<ggc-version>/rootfs/upper,workdir=/greengrass/ggc/packages/<ggc-version>/rootfs/work\": terlalu banyak tingkatan tautan simbolik"}

Solusi: Anda mungkin melihat kesalahan ini di `runtime.log` ketika file AWS IoT Greengrass perangkat lunak core tidak mulai. Masalah ini mungkin lebih umum pada sistem operasi Debian.

Untuk mengatasi masalah ini, lakukan solusi berikut:

1. Upgrade AWS IoT Greengrass perangkat lunak Core ke v1.9.3 atau yang lebih baru. Ini akan secara otomatis mengatasi masalah ini.
2. Jika Anda masih mendapatkan error ini setelah Anda memperbarui AWS IoT Greengrass perangkat lunak Core, mengatur `system.useOverlayWithTmpfs` properti ke `true` di file config.json ini.

   Example Contoh

   ```
   {
     "system": {
       "useOverlayWithTmpfs": true
     },
     "coreThing": {
       "caPath": "root-ca.pem",
       "certPath": "cloud.pem.crt",
       "keyPath": "cloud.pem.key",
       ...
     },
     ...
   }
   ```

> **Note**
>
> Versi perangkat lunak core AWS IoT Greengrass Anda ditampilkan dalam pesan error. Untuk menemukan versi kernel Linux Anda, jalankan `uname -r`.

Error: [DEBUG]-Gagal untuk mendapatkan rute. Membuang pesan.

Solusi: Periksa langganan di grup Anda dan pastikan bahwa langganan tercantum dalam `[DEBUG]` pesan yang ada.

Error: [Errno 24] Terlalu banyak membuka <lambda-function>, [Errno 24] Terlalu banyak membuka file

Solusi: Anda mungkin melihat kesalahan ini di berkas log fungsi Lambda Anda jika fungsi instantiates `StreamManagerClient` dalam fungsi handler. Kami merekomendasikan bahwa Anda membuat klien di luar handler. Untuk informasi selengkapnya, lihat the section called “Gunakan StreamManagerClient untuk bekerja dengan aliran”.

Kesalahan: server ds gagal mulai mendengarkan soket: dengarkan unix <ggc-path>/ggc/socket/greengrass_ipc.sock: bind: argumen tidak valid

Solusi: Anda mungkin melihat kesalahan ini ketika perangkat lunak AWS IoT Greengrass Core tidak dimulai. Kesalahan ini terjadi ketika perangkat lunak AWS IoT Greengrass Core diinstal ke folder dengan jalur file yang panjang. Instal ulang perangkat lunak AWS IoT Greengrass Core ke folder dengan jalur file yang memiliki kurang dari 79 byte, jika Anda tidak menggunakan direktori tulis, atau 83 byte, jika Anda menggunakan direktori tulis.

[INFO] (Mesin Fotokopi) aws.greengrass. StreamManager: stdout. Disebabkan oleh: com.fasterxml.jackson.databind. JsonMappingException: Instan melebihi minimum atau maksimum instan

Saat Anda memutakhirkan perangkat lunak AWS IoT Greengrass inti ke v1.11.3, Anda mungkin melihat kesalahan berikut di log pengelola aliran jika pengelola aliran gagal memulai.

```
2021-07-16T00:54:58.568Z [INFO] (Copier) aws.greengrass.StreamManager:
 stdout. Caused by: com.fasterxml.jackson.databind.JsonMappingException:
 Instant exceeds minimum or maximum instant (through reference chain:
 com.amazonaws.iot.greengrass.streammanager.export.PersistedSuccessExportStatesV1["lastExportTi
 {scriptName=services.aws.greengrass.StreamManager.lifecycle.startup.script,
 serviceName=aws.greengrass.StreamManager, currentState=STARTING}
2021-07-16T00:54:58.579Z [INFO] (Copier) aws.greengrass.StreamManager: stdout.
 Caused by: java.time.DateTimeException: Instant exceeds minimum or maximum instant.
 {scriptName=services.aws.greengrass.StreamManager.lifecycle.startup.script,
 serviceName=aws.greengrass.StreamManager, currentState=STARTING}
```

Jika Anda menggunakan versi perangkat lunak AWS IoT Greengrass inti yang lebih lama dari v1.11.3, dan Anda ingin meningkatkan ke versi yang lebih baru, gunakan pembaruan OTA untuk meningkatkan ke v1.11.4.

GPG error: https://dnw9lb6lzp2d8.cloudfront.net stable InRelease: The following signatures were invalid: EXPKEYSIG 68D644ABD2327D47 AWS Greengrass Master Key

Ketika Anda menjalankan perangkat `apt update` di mana Anda menginstal perangkat lunak AWS IoT Greengrass inti dari repositori APT, Anda mungkin melihat kesalahan berikut.

```
Err:4 https://dnw9lb6lzp2d8.cloudfront.net stable InRelease
  The following signatures were invalid: EXPKEYSIG 68D644ABD2327D47 AWS Greengrass
 Master Key
Reading package lists... Done
W: GPG error: https://dnw9lb6lzp2d8.cloudfront.net stable InRelease: The following
 signatures were invalid: EXPKEYSIG 68D644ABD2327D47 AWS Greengrass Master Key
```

Kesalahan ini terjadi karena AWS IoT Greengrass tidak lagi menawarkan opsi untuk menginstal atau memperbarui perangkat lunak AWS IoT Greengrass inti dari repositori APT. Agar berhasil dijalankan`apt update`, hapus AWS IoT Greengrass repositori dari daftar sumber perangkat.

```
sudo rm /etc/apt/sources.list.d/greengrass.list
sudo apt update
```

Masalah deployment

Gunakan informasi berikut untuk membantu memecahkan masalah deployment.

Masalah

- Deployment Anda saat ini tidak bekerja dan Anda ingin kembali ke deployment kerja sebelumnya.
- Anda melihat 403 error terlarang pada deployment di catatan.
- Terjadi ConcurrentDeployment kesalahan saat Anda menjalankan perintah create-deployment untuk pertama kalinya.
- Error: Greengrass tidak berwenang untuk menganggap peran layanan yang terkait dengan akun ini, atau error: Gagal: peran layanan TES tidak terkait dengan akun ini.
- Error: tidak dapat menjalankan langkah pengunduhan dalam deployment. error saat mengunduh: error saat mengunduh file definisi Grup:... x509: sertifikat telah kedaluwarsa atau belum valid
- Deployment tidak selesai.
- Kesalahan: Tidak dapat menemukan executable java atau java8, atau kesalahan: Penerapan <deployment-id>tipe NewDeployment untuk grup <group-id>gagal kesalahan: pekerja dengan <worker-id>gagal menginisialisasi dengan alasan Versi Java yang diinstal harus lebih besar dari atau sama dengan 8
- Deployment tidak selesai, dan runtime.catatan berisi beberapa entri "tunggu 1 detik untuk kontainer untuk berhenti".
- Deployment tidak selesai, dan runtime.log berisi "[KESALAHAN]-Greengrass deployment error: gagal melaporkan status deployment kembali ke cloud {"deploymentId": "<deployment-id>", "errorString": "Failed to initiate PUT, endpoint: https://<deployment-status>, error: Put https://<deployment-status>: proxyconnect tcp: x509: certificate signed by unknown authority"}"
- <path>Kesalahan: <deployment-id>Penerapan tipe NewDeployment untuk grup <group-id>gagal kesalahan: Kesalahan saat memproses. konfigurasi grup tidak valid: 112 atau [119 0] tidak memiliki izin rw pada file:.
- Kesalahan: < list-of-function-arns > dikonfigurasi untuk berjalan sebagai root tetapi Greengrass tidak dikonfigurasi untuk menjalankan fungsi Lambda dengan izin root.
- Kesalahan: <deployment-id>Penerapan tipe NewDeployment untuk grup <group-id>gagal kesalahan: Kesalahan penerapan Greengrass: tidak dapat menjalankan langkah unduhan dalam

penerapan. kesalahan saat memproses: tidak dapat memuat file grup yang diunduh: tidak dapat menemukan UID berdasarkan nama pengguna, UserName: ggc_user: user: unknown user ggc_user.

- Error: [ERROR]-error eksekusi waktu aktif: tidak dapat meluncurkan kontainer lambda. {"ErrorString": "gagal menginisialisasi pemasangan kontainer: gagal untuk menutupi root greengrass di overlay dir atas: gagal membuat perangkat mask pada direktori <ggc-path>: file ada"}
- Kesalahan: Penerapan <deployment-id>tipe NewDeployment untuk grup <group-id>gagal kesalahan: proses mulai gagal: container_linux.go: 259: memulai proses kontainer menyebabkan “process_linux.go: 250: menjalankan exec setns process for init caused\" wait: no child processes\ "”.
- <host-prefix>Kesalahan: [PERINGATAN] -MQTT [klien] panggil tcp: lookup -ats.iot. <region>.amazonaws.com: tidak ada host seperti itu... [ERROR]-Greengrass deployment error: gagal melaporkan status deployment kembali ke cloud ... net/http: permintaan dibatalkan saat menunggu koneksi (Client.Timeout terlampaui saat menunggu header)

Deployment Anda saat ini tidak bekerja dan Anda ingin kembali ke deployment kerja sebelumnya.

Solusi: Gunakan konsol AWS IoT atau API AWS IoT Greengrass untuk men-deploy ulang deployment kerja sebelumnya. Ini men-deploy versi grup yang sesuai ke perangkat core Anda.

Untuk men-deploy ulang deployment (konsol)

1. Pada halaman konfigurasi grup, pilih tab Deployment. Halaman ini menampilkan riwayat penerapan untuk grup, termasuk tanggal dan waktu, versi grup, dan status setiap upaya deployment.
2. Temukan baris yang berisi deployment Anda ingin men-deploy ulang. Pilih penyebaran yang ingin Anda gunakan kembali dan pilih Redeploy.

Untuk men-deploy ulang deployment (CLI)

1. Gunakan ListDeploymentsuntuk menemukan ID penerapan yang ingin Anda gunakan kembali. Sebagai contoh:

   ```
   aws greengrass list-deployments --group-id 74d0b623-c2f2-4cad-9acc-ef92f61fcaf7
   ```

   Perintah mengembalikan daftar deployment untuk grup.

   ```
   {
       "Deployments": [
           {
               "DeploymentId": "8d179428-f617-4a77-8a0c-3d61fb8446a6",
               "DeploymentType": "NewDeployment",
               "GroupArn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
   groups/74d0b623-c2f2-4cad-9acc-ef92f61fcaf7/versions/8dd1d899-4ac9-4f5d-
   afe4-22de086efc62",
               "CreatedAt": "2019-07-01T20:56:49.641Z"
           },
           {
               "DeploymentId": "f8e4c455-8ac4-453a-8252-512dc3e9c596",
               "DeploymentType": "NewDeployment",
               "GroupArn": "arn:aws:greengrass:us-west-2::123456789012:/greengrass/
   groups/74d0b623-c2f2-4cad-9acc-ef92f61fcaf7/versions/4ad66e5d-3808-446b-940a-
   b1a788898382",
               "CreatedAt": "2019-07-01T20:41:47.048Z"
           },
           {
               "DeploymentId": "e4aca044-bbd8-41b4-b697-930ca7c40f3e",
               "DeploymentType": "NewDeployment",
   ```

```
            "GroupArn": "arn:aws:greengrass:us-west-2::123456789012:/greengrass/
groups/74d0b623-c2f2-4cad-9acc-ef92f61fcaf7/versions/1f3870b6-850e-4c97-8018-
c872e17b235b",
            "CreatedAt": "2019-06-18T15:16:02.965Z"
        }
    ]
}
```

> **Note**
>
> Ini AWS CLI menggunakan nilai contoh untuk ID grup dan deployment. Ketika Anda menjalankan perintah, pastikan untuk mengganti nilai contoh.

2. Gunakan CreateDeploymentuntuk menerapkan kembali penyebaran target. Atur jenis deployment ke `Redeployment`. Sebagai contoh:

```
aws greengrass create-deployment --deployment-type Redeployment \
  --group-id 74d0b623-c2f2-4cad-9acc-ef92f61fcaf7 \
  --deployment-id f8e4c455-8ac4-453a-8252-512dc3e9c596
```

Perintah mengembalikan ARN dan ID dari deployment baru.

```
{
    "DeploymentId": "f9ed02b7-c28e-4df6-83b1-e9553ddd0fc2",
    "DeploymentArn": "arn:aws:greengrass:us-west-2::123456789012:/greengrass/
groups/74d0b623-c2f2-4cad-9acc-ef92f61fcaf7/deployments/f9ed02b7-c28e-4df6-83b1-
e9553ddd0fc2"
}
```

3. Gunakan GetDeploymentStatusuntuk mendapatkan status penyebaran.

Anda melihat 403 error terlarang pada deployment di catatan.

Solusi: Pastikan kebijakan AWS IoT Greengrass core di cloud termasuk `"greengrass:*"` sebagai tindakan yang diizinkan.

Terjadi ConcurrentDeployment kesalahan saat Anda menjalankan perintah create-deployment untuk pertama kalinya.

Solusi: Sebuah deployment mungkin sedang berjalan. Anda dapat menjalankan get-deployment-statusuntuk melihat apakah penerapan telah dibuat. Jika tidak, coba membuat deployment lagi.

Error: Greengrass tidak berwenang untuk menganggap peran layanan yang terkait dengan akun ini, atau error: Gagal: peran layanan TES tidak terkait dengan akun ini.

Solusi: Anda mungkin melihat kesalahan ini ketika deployment gagal. Periksa apakah peran layanan Greengrass dikaitkan dengan Akun AWS Anda saat ini Wilayah AWS. Untuk informasi lebih lanjut, lihat the section called “Mengelola peran layanan (CLI)” atau the section called “Mengelola peran layanan (konsol)”.

Error: tidak dapat menjalankan langkah pengunduhan dalam deployment. error saat mengunduh: error saat mengunduh file definisi Grup:... x509: sertifikat telah kedaluwarsa atau belum valid

Solusi: Anda mungkin melihat kesalahan ini di `runtime.log` ketika deployment gagal. Jika Anda menerima `Deployment failed` error yang berisi pesan `x509: certificate has expired or is not yet valid`, periksa jam perangkat. Sertifikat TLS dan X.509 memberikan dasar yang aman untuk membangun sistem IoT, tetapi mereka memerlukan waktu yang akurat pada server dan klien. Perangkat IoT harus memiliki waktu yang benar (dalam waktu 15 menit) sebelum mereka mencoba untuk terhubung ke AWS IoT Greengrass atau layanan TLS lain yang menggunakan sertifikat server. Untuk informasi selengkapnya, lihat Menggunakan Waktu Perangkat untuk Validasi AWS IoT Sertifikat Server pada Internet of Things pada AWS Blog Resmi.

Deployment tidak selesai.

Solusi: Lakukan hal berikut:

- Pastikan bahwa AWS IoT Greengrass daemon berjalan pada perangkat core Anda. Di terminal perangkat core Anda, jalankan perintah berikut untuk memeriksa apakah daemon sedang berjalan dan memulainya, jika diperlukan.

  1. Untuk memeriksa apakah daemon sedang berjalan:

     ```
     ps aux | grep -E 'greengrass.*daemon'
     ```

     Jika outputnya berisi entri `root` untuk `/greengrass/ggc/packages/1.11.6/bin/daemon`, maka daemon sedang berjalan.

     Versi di jalur tergantung pada versi perangkat lunak AWS IoT Greengrass core yang diinstal pada perangkat core Anda.

  2. Untuk mulai daemon:

     ```
     cd /greengrass/ggc/core/
     sudo ./greengrassd start
     ```

- Pastikan bahwa perangkat core terhubung dan titik akhir koneksi core terkonfigurasi dengan benar.

Kesalahan: Tidak dapat menemukan executable java atau java8, atau kesalahan: Penerapan <deployment-id>tipe NewDeployment untuk grup <group-id>gagal kesalahan: pekerja dengan <worker-id>gagal menginisialisasi dengan alasan Versi Java yang diinstal harus lebih besar dari atau sama dengan 8

Solusi: Jika pengelola aliran diaktifkan untuk AWS IoT Greengrass core, Anda harus menginstal waktu aktif Java 8 pada perangkat core sebelum Anda men-deploy grup. Untuk informasi selengkapnya, lihat persyaratan untuk pengelola aliran. Pengelola aliran diaktifkan secara default ketika Anda menggunakan Pembuatan Grup default alur kerja di konsol AWS IoT tersebut untuk membuat grup.

Atau, nonaktifkan pengelola aliran dan kemudian men-deploy grup. Untuk informasi selengkapnya, lihat the section called “Konfigurasikan pengaturan (konsol)”.

Deployment tidak selesai, dan runtime.catatan berisi beberapa entri "tunggu 1 detik untuk kontainer untuk berhenti".

Solusi: Jalankan perintah berikut di terminal perangkat core Anda untuk me-restart AWS IoT Greengrass daemon.

```
cd /greengrass/ggc/core/
sudo ./greengrassd stop
sudo ./greengrassd start
```

Deployment tidak selesai, dan **`runtime.log`** berisi "[KESALAHAN]- Greengrass deployment error: gagal melaporkan status deployment kembali ke cloud {"deploymentId": "<deployment-id>", "errorString": "Failed to initiate PUT, endpoint: https://<deployment-status>, error: Put https://<deployment-status>: proxyconnect tcp: x509: certificate signed by unknown authority"}"

Solusi: Anda mungkin melihat kesalahan ini di `runtime.log` ketika Greengrass core dikonfigurasi untuk menggunakan koneksi proksi HTTPS dan rantai sertifikat server proksi tidak dipercaya pada sistem. Untuk mencoba mengatasi masalah ini, tambahkan rantai sertifikat ke root sertifikat CA. Greengrass core menambahkan sertifikat dari file ini ke kolam sertifikat yang digunakan untuk otentikasi koneksi TLS di HTTPS dan MQTT dengan AWS IoT Greengrass.

Contoh berikut menunjukkan sertifikat CA server proksi ditambahkan ke file root sertifikat CA:

```
# My proxy CA
-----BEGIN CERTIFICATE-----
MIIEFTCCAv2gAwIQWgIVAMHSAzWG/5YVRYtRQOxXUTEpHuEmApzGCSqGSIb3DQEK
\nCwUAhuL9MQswCQwJVUzEPMAVUzEYMBYGA1UECgwP1hem9uLmNvbSBJbmMuMRww
... content of proxy CA certificate ...
+vHIRlt0e5JAm5\noTIZGoFbK82A0/nO7f/t5PSIDAim9V3Gc3pSXxCCAQoFYnui
GaPUlGk1gCE84a0X\n7Rp/lND/PuMZ/s8YjlkY2NmYmNjMCAXDTE5MTEyN2cM216
gJMIADggEPADf2/m45hzEXAMPLE=
-----END CERTIFICATE-----

# Amazon Root CA 1
```

```
-----BEGIN CERTIFICATE-----
MIIDQTCCAimgF6AwIBAgITBmyfz/5mjAo54vB4ikPmljZKyjANJmApzyMZFo6qBg
ADA5MQswCQYDVQQGEwJVUzEPMA0tMVT8QtPHRh8jrdkGA1UEChMGDV3QQDExBBKW
... content of root CA certificate ...
o/ufQJQWUCyziar1hem9uMRkwFwYVPSHCb2XV4cdFyQzR1KldZwgJcIQ6XUDgHaa
5MsI+yMRQ+hDaXJiobldXgjUka642M4UwtBV8oK2xJNDd2ZhwLnoQdeXeGADKkpy
rqXRfKoQnoZsG4q5WTP46EXAMPLE
-----END CERTIFICATE-----
```

Secara default, file root sertifikat CA terletak di */greengrass-root*/certs/root.ca.pem. Untuk menemukan lokasi di perangkat core, periksa properti `crypto.caPath` di config.json.

> **Note**
>
> *greengrass-root* mewakili jalur di mana AWS IoT Greengrass perangkat lunak core diinstal pada perangkat Anda. Biasanya, adalah direktori `/greengrass` ini.

<path>Kesalahan: <deployment-id>Penerapan tipe NewDeployment untuk grup <group-id>gagal kesalahan: Kesalahan saat memproses. konfigurasi grup tidak valid: 112 atau [119 0] tidak memiliki izin rw pada file:.

Solusi: Pastikan bahwa grup pemilik *<path>* direktori memiliki izin baca dan tulis ke direktori.

Kesalahan: < list-of-function-arns > dikonfigurasi untuk berjalan sebagai root tetapi Greengrass tidak dikonfigurasi untuk menjalankan fungsi Lambda dengan izin root.

Solusi: Anda mungkin melihat kesalahan ini di `runtime.log` ketika deployment gagal. Pastikan bahwa Anda telah mengonfigurasi AWS IoT Greengrass untuk mengizinkan fungsi Lambda berjalan dengan izin root. Baik mengubah nilai `allowFunctionsToRunAsRoot` masuk `greengrass_root/config/config.json` ke `yes` atau mngubah fungsi Lambda agar berjalan sebagai pengguna/group lain. Untuk informasi selengkapnya, lihat the section called “Menjalankan fungsi Lambda sebagai root”.

Kesalahan: <deployment-id>Penerapan tipe NewDeployment untuk grup <group-id>gagal kesalahan: Kesalahan penerapan Greengrass: tidak dapat menjalankan langkah unduhan dalam penerapan. kesalahan saat memproses: tidak dapat memuat file grup yang diunduh: tidak dapat menemukan UID berdasarkan nama pengguna, UserName: ggc_user: user: unknown user ggc_user.

Solusi: Jika identitas akses default dari AWS IoT Greengrass grup menggunakan akun sistem standar, `ggc_user` pengguna dan `ggc_group` grup harus hadir pada perangkat. Untuk petunjuk yang menunjukkan cara menambahkan pengguna dan grup, lihat langkah ini. Pastikan untuk memasukkan nama persis seperti yang ditunjukkan.

Error: [ERROR]-error eksekusi waktu aktif: tidak dapat meluncurkan kontainer lambda. {"ErrorString": "gagal menginisialisasi pemasangan kontainer: gagal untuk menutupi root greengrass di overlay dir atas: gagal membuat perangkat mask pada direktori <ggc-path>: file ada"}

Solusi: Anda mungkin melihat kesalahan ini di `runtime.log` ketika deployment gagal. error ini terjadi jika fungsi Lambda dalam grup Greengrass tidak dapat mengakses direktori `/usr` dalam sistem file core. Untuk mengatasi masalah ini, tambahkan sumber daya volume lokal ke grup dan kemudian men-deploy grup. Sumber daya harus:

- Tentukan `/usr` sebagai Jalur sumber dan Jalur tujuan.
- Secara otomatis menambahkan izin grup OS dari grup Linux yang memiliki sumber daya.
- Berafiliasi dengan fungsi Lambda dan mengizinkan akses hanya baca.

Kesalahan: Penerapan <deployment-id>tipe NewDeployment untuk grup <group-id>gagal kesalahan: proses mulai gagal: container_linux.go: 259: memulai proses kontainer menyebabkan “process_linux.go: 250: menjalankan exec setns process for init caused\" wait: no child processes\ "”.

Solusi: Anda mungkin melihat kesalahan ini ketika deployment gagal. Coba lagi deployment.

<host-prefix>Kesalahan: [PERINGATAN] -MQTT [klien] panggil tcp: lookup -ats.iot. <region>.amazonaws.com: tidak ada host seperti itu... [ERROR]-Greengrass deployment error: gagal melaporkan status deployment kembali ke cloud ... net/http: permintaan dibatalkan saat menunggu koneksi (Client.Timeout terlampaui saat menunggu header)

Solusi: Anda mungkin melihat kesalahan ini jika Anda menggunakan `systemd-resolved`, yang memungkinkan `DNSSEC` pengaturan secara default. Akibatnya, banyak domain publik tidak dikenali. Upaya untuk mencapai titik akhir AWS IoT Greengrass gagal untuk menemukan host, sehingga penyebaran Anda tetap dalam status `In Progress` ini.

Anda dapat menggunakan perintah berikut dan output untuk menguji masalah ini. Ganti placeholder *`wilayah`* di titik akhir dengan Wilayah AWS.

```
$ ping greengrass-ats.iot.region.amazonaws.com
ping: greengrass-ats.iot.region.amazonaws.com: Name or service not known
```

```
$ systemd-resolve greengrass-ats.iot.region.amazonaws.com
greengrass-ats.iot.region.amazonaws.com: resolve call failed: DNSSEC validation failed:
 failed-auxiliary
```

Satu penyelesaian yang mungkin adalah menonaktifkan `DNSSEC`. Ketika `DNSSEC` adalah `false`, pencarian DNS tidak `DNSSEC` divalidasi. Untuk informasi selengkapnya, lihat masalah diketahui untuk `systemd`.

1. Tambahkan `DNSSEC=false` ke `/etc/systemd/resolved.conf`.

2. Mulai ulang `systemd-resolved`.

Untuk informasi tentang `resolved.conf` dan `DNSSEC`, jalankan `man resolved.conf` di terminal Anda.

Buat grup dan buat masalah fungsi

Gunakan informasi berikut untuk membantu memecahkan masalah dengan membuat AWS IoT Greengrass grup atau fungsi Greengrass Lambda.

Masalah

- Kesalahan: Konfigurasi IsolationMode " Anda untuk grup tidak valid.
- Kesalahan: Konfigurasi 'IsolationMode' Anda untuk fungsi dengan arn <function-arn>tidak valid.
- Kesalahan: MemorySize konfigurasi untuk fungsi dengan arn <function-arn>tidak diperbolehkan di IsolationMode =NoContainer.
- Kesalahan: Akses konfigurasi Sysfs untuk fungsi dengan arn <function-arn>tidak diperbolehkan di =. IsolationMode NoContainer
- Kesalahan: MemorySize konfigurasi untuk fungsi dengan arn <function-arn>diperlukan di IsolationMode =GreengrassContainer.
- Kesalahan: Fungsi <function-arn>mengacu pada sumber daya tipe <resource-type>yang tidak diperbolehkan di IsolationMode =NoContainer.
- Error: Konfigurasi eksekusi untuk fungsi dengan arn <function-arn> tidak diizinkan.

Kesalahan: Konfigurasi IsolationMode " Anda untuk grup tidak valid.

Solusi: error ini terjadi ketika `IsolationMode` nilai dalam `DefaultConfig` dari `function-definition-version` tidak didukung. Nilai yang didukung adalah `GreengrassContainer` dan `NoContainer`.

Kesalahan: Konfigurasi 'IsolationMode' Anda untuk fungsi dengan arn <function-arn>tidak valid.

Solusi: error ini terjadi ketika `IsolationMode` nilai <function-arn> dalam `function-definition-version` tidak didukung. Nilai yang didukung adalah `GreengrassContainer` dan `NoContainer`.

Kesalahan: MemorySize konfigurasi untuk fungsi dengan arn <function-arn>tidak diperbolehkan di IsolationMode =NoContainer.

Solusi: error ini terjadi saat Anda menentukan `MemorySize` nilai dan Anda memilih untuk menjalankan tanpa kontainerisasi. Fungsi Lambda yang dijalankan tanpa kontainerisasi tidak dapat memiliki batas memori. Anda dapat menghapus batas atau Anda dapat mengubah fungsi Lambda untuk menjalankan di AWS IoT Greengrass kontainer.

Kesalahan: Akses konfigurasi Sysfs untuk fungsi dengan arn <function-arn>tidak diperbolehkan di =. IsolationMode NoContainer

Solusi: Error ini terjadi saat Anda menentukan `true` untuk `AccessSysfs` dan Anda memilih untuk menjalankan tanpa kontainerisasi. Fungsi Lambda berjalan tanpa containerization harus memiliki kode mereka diperbarui untuk mengakses sistem file secara langsung dan tidak dapat menggunakan `AccessSysfs`. Anda dapat menentukan nilai `false` untuk `AccessSysfs` atau Anda dapat mengubah fungsi Lambda agar berjalan di AWS IoT Greengrass kontainer.

Kesalahan: MemorySize konfigurasi untuk fungsi dengan arn <function-arn>diperlukan di IsolationMode =GreengrassContainer.

Solusi: Error ini terjadi karena Anda tidak menentukan `MemorySize` batas untuk fungsi Lambda yang Anda jalankan di AWS IoT Greengrass kontainer. Tentukan nilai `MemorySize` untuk mengatasi error.

Kesalahan: Fungsi <function-arn>mengacu pada sumber daya tipe <resource-type>yang tidak diperbolehkan di IsolationMode =NoContainer.

Solusi: Anda tidak dapat mengakses `Local.Device`, `Local.Volume`, `ML_Model.SageMaker.Job`, `ML_Model.S3_Object`, atau `S3_Object.Generic_Archive` jenis sumber daya ketika Anda menjalankan fungsi Lambda tanpa kontainerisasi. Jika Anda membutuhkan jenis sumber daya tersebut, Anda harus menjalankan di AWS IoT Greengrass kontainer. Anda juga dapat mengakses perangkat lokal secara langsung ketika berjalan tanpa kontainerisasi dengan mengubah kode di fungsi Lambda Anda.

Error: Konfigurasi eksekusi untuk fungsi dengan arn <function-arn> tidak diizinkan.

Solusi: Error ini terjadi saat Anda membuat fungsi sistem Lambda dengan `GGIPDetector` atau `GGCloudSpooler` dan Anda tentukan `IsolationMode` atau `RunAs` konfigurasi. Anda harus menghilangkan `Execution` parameter untuk sistem fungsi Lambda ini.

Masalah Penemuan

Informasi berikut dapat membantu Anda memecahkan masalah dengan AWS IoT Greengrass layanan Penemuan.

Masalah

- Error: Perangkat adalah anggota dari terlalu banyak grup, perangkat mungkin tidak berada di lebih dari 10 grup

Error: Perangkat adalah anggota dari terlalu banyak grup, perangkat mungkin tidak berada di lebih dari 10 grup

Solusi: Ini adalah batasan yang diketahui. Perangkat klien dapat menjadi anggota hingga 10 grup.

Masalah sumber daya machine learning

Gunakan informasi berikut untuk membantu memecahkan masalah dengan sumber daya machine learning.

Masalah

- InvalidML ModelOwner - GroupOwnerSetting disediakan dalam sumber daya model ML, tetapi GroupOwner atau GroupPermission tidak ada
- NoContainer fungsi tidak dapat mengonfigurasi izin saat melampirkan sumber daya Machine Learning. <function-arn>mengacu pada sumber daya Machine Learnin <resource-id>dengan izin <ro/rw> dalam kebijakan akses sumber daya.
- Fungsi <function-arn>mengacu pada sumber daya Machine Learning <resource-id>dengan izin yang hilang di keduanya ResourceAccessPolicy dan sumber daya OwnerSetting.
- Fungsi <function-arn>mengacu pada sumber daya Machine Learning <resource-id>dengan izin\ "rw\", sedangkan pengaturan pemilik sumber daya GroupPermission hanya mengizinkan\ "ro\".
- NoContainer Fungsi <function-arn>mengacu pada sumber daya jalur tujuan bersarang.
- Lambda <function-arn> mendapatkan akses ke sumber daya <resource-id> dengan berbagi id pemilik grup yang sama

InvalidML ModelOwner - GroupOwnerSetting disediakan dalam sumber daya model ML, tetapi GroupOwner atau GroupPermission tidak ada

Solusi: Anda menerima kesalahan ini jika sumber pembelajaran mesin berisi ResourceDownloadOwnerSettingobjek tetapi diperlukan `GroupOwner` atau `GroupPermission` properti tidak ditentukan. Untuk mengatasi masalah ini, tentukan properti yang hilang.

NoContainer fungsi tidak dapat mengonfigurasi izin saat melampirkan sumber daya Machine Learning. <function-arn>mengacu pada sumber daya Machine Learnin <resource-id>dengan izin <ro/rw> dalam kebijakan akses sumber daya.

Solusi: Anda menerima error ini jika fungsi Lambda non-containerized menentukan tingkat fungsi izin untuk sumber daya machine learning. Fungsi non-wadah harus mewarisi izin dari izin pemilik sumber daya yang ditetapkan pada sumber daya machine learning. Untuk mengatasi masalah ini, pilih untuk mewarisi izin pemilik sumber daya (konsol) atau menghapus izin dari fungsi Lambda sumber daya kebijakan akses (API).

Fungsi <function-arn>mengacu pada sumber daya Machine Learning <resource-id>dengan izin yang hilang di keduanya ResourceAccessPolicy dan sumber daya OwnerSetting.

Solusi: Anda menerima kesa ini jika izin untuk sumber daya machine learning tidak dikonfigurasi untuk fungsi Lambda terlampir atau sumber daya. Untuk mengatasi masalah ini, konfigurasikan izin di ResourceAccessPolicyproperti untuk fungsi Lambda atau properti untuk OwnerSettingsumber daya.

Fungsi <function-arn>mengacu pada sumber daya Machine Learning <resource-id>dengan izin\ "rw\”, sedangkan pengaturan pemilik sumber daya GroupPermission hanya mengizinkan\ "ro\”.

Solusi: Anda menerima error ini jika izin akses yang ditetapkan untuk fungsi Lambda terlampir melebihi izin pemilik sumber daya yang ditetapkan untuk sumber daya machine learning. Untuk mengatasi masalah ini, tetapkan izin yang lebih ketat untuk fungsi Lambda atau kurang membatasi izin untuk pemilik sumber daya.

NoContainer Fungsi <function-arn>mengacu pada sumber daya jalur tujuan bersarang.

Solusi: Anda menerima error ini jika beberapa sumber daya machine learning yang terlampir pada fungsi Lambda non-containerized menggunakan lintasan tujuan yang sama atau lintasan tujuan bersarang. Untuk mengatasi masalah ini, tentukan jalur tujuan terpisah untuk sumber daya.

Lambda <function-arn> mendapatkan akses ke sumber daya <resource-id> dengan berbagi id pemilik grup yang sama

Solusi: Anda menerima keslaahan ini di `runtime.log` jika kelompok OS yang sama ditentukan sebagai fungsi Lambda Jalankan sebagai identitas dan pemilik sumber daya untuk sumber daya machine learning, tetapi sumber daya tidak terlampir pada fungsi Lambda. Konfigurasi ini memberikan fungsi Lambda izin implisit yang dapat digunakan untuk mengakses sumber daya tanpa AWS IoT Greengrass otorisasi.

Untuk mengatasi masalah ini, gunakan grup OS yang berbeda untuk salah satu properti atau melampirkan sumber daya machine learning untuk fungsi Lambda.

AWS IoT Greengrass core dalam masalah Docker

Gunakan informasi berikut untuk membantu memecahkan masalah dengan menjalankan AWS IoT Greengrass core di kontainer Docker.

Masalah

- Kesalahan: Opsi tidak diketahui: -no-include-email.
- Peringatan: IPv4 dinonaktifkan. Jaringan tidak akan bekerja.
- Error: Firewall memblokir file berbagi antara windows dan kontainer.
- Kesalahan: Terjadi kesalahan (AccessDeniedException) saat memanggil GetAuthorizationToken operasi: User: arn:aws:iam: ::user/ <account-id><user-name>tidak diizinkan untuk melakukan: ecr: on resource: * GetAuthorizationToken
- Error: Tidak dapat membuat kontainer untuk greengrass layanan: konflik. Nama kontainer “/aws-iot-greengrass" sudah digunakan.

- Error: [FATAL]-Gagal untuk mereset thread mount namespace karena error tak terduga: "operasi tidak diizinkan". Untuk menjaga konsistensi, GGC akan macet dan harus dimulai ulang secara manual.

Kesalahan: Opsi tidak diketahui: -no-include-email.

Solusi: Kesalahan ini dapat terjadi ketika Anda menjalankan perintah `aws ecr get-login` ini. Pastikan Anda memiliki versi AWS CLI terbaru yang terinstal (misalnya, jalankan: `pip install awscli --upgrade --user`). Jika Anda menggunakan Windows dan Anda menginstal CLI menggunakan MSI installer, Anda harus mengulangi proses instalasi. Untuk informasi lebih lanjut, lihat Menginstal AWS Command Line Interface di Microsoft Windows dalam AWS Command Line Interface Panduan Pengguna.

Peringatan: IPv4 dinonaktifkan. Jaringan tidak akan bekerja.

Solusi: Anda mungkin menerima peringatan ini atau pesan serupa ketika menjalankan AWS IoT Greengrass pada komputer Linux. Aktifkan penerusan jaringan IPv4 seperti yang dijelaskan dalam Langkah. AWS IoT Greengrass cloud deployment dan komunikasi MQTT tidak bekerja ketika penerusan IPv4 tidak diaktifkan. Untuk informasi lebih lanjut, lihat Mengonfigurasi parameter kernel namespace (sysctls) pada ketika waktu aktif dalam dokumentasi Docker.

Error: Firewall memblokir file berbagi antara windows dan kontainer.

Solusi: Anda mungkin menerima error ini atau pesan `Firewall Detected` ketika menjalankan Docker di komputer Windows. Hal ini juga dapat terjadi jika Anda masuk pada jaringan pribadi virtual (VPN) dan pengaturan jaringan Anda mencegah drive berbagi untuk dipasang. Dalam situasi itu, matikan VPN dan jalankan kembali kontainer Docker.

Kesalahan: Terjadi kesalahan (AccessDeniedException) saat memanggil GetAuthorizationToken operasi: User: arn:aws:iam: ::user/ <account-id><user-name>tidak diizinkan untuk melakukan: ecr: on resource: * GetAuthorizationToken

Anda mungkin menerima kesalahan ini saat menjalankan `aws ecr get-login-password` jika Anda tidak memiliki izin yang memadai untuk mengakses repositori Amazon ECR. Untuk informasi selengkapnya, lihat Contoh Kebijakan Repositori Amazon ECR dan Mengakses Satu Repositori Amazon ECR di Panduan Pengguna Amazon ECR.

Error: Tidak dapat membuat kontainer untuk greengrass layanan: konflik. Nama kontainer “/aws-iot-greengrass" sudah digunakan.

Solusi: Hal ini dapat terjadi ketika nama kontainer yang digunakan oleh kontainer yang lebih tua. Untuk mengatasi masalah ini, jalankan perintah berikut untuk menghapus kontainer Docker lama:

```
docker rm -f $(docker ps -a -q -f "name=aws-iot-greengrass")
```

Error: [FATAL]-Gagal untuk mereset thread mount namespace karena error tak terduga: "operasi tidak diizinkan". Untuk menjaga konsistensi, GGC akan macet dan harus dimulai ulang secara manual.

Solusi: error ini di `runtime.log` dapat terjadi ketika Anda mencoba untuk men-deploy `GreengrassContainer` fungsi Lambda ke sebuah AWS IoT Greengrass core berjalan dalam kontainer Docker. Saat ini, hanya `NoContainer` fungsi Lambda dapat dikerahkan ke kontainer Greengrass Docker.

Untuk mengatasi masalah ini, pastikan bahwa semua fungsi Lambda berada di mode `NoContainer` ini dan mulai deployment baru. Kemudian, ketika memulai kontainer, jangan bind-mount direktori `deployment` yang ada ke AWS IoT Greengrass kontainer Docker core. Sebaliknya, buat kosong `deployment` di tempatnya dan bind-mount yang di kontainer Docker. Hal ini mengzinkan kontainer Docker baru untuk menerima deployment terbaru dengan fungsi Lambda berjalan di `NoContainer` mode.

Untuk informasi selengkapnya, lihat the section called “Jalankan AWS IoT Greengrass di kontainer Docker”.

Pemecahan masalah dengan catatan

Anda dapat mengonfigurasi pengaturan logging untuk grup Greengrass, seperti apakah akan mengirim log CloudWatch ke Log, menyimpan log pada sistem file lokal, atau keduanya. Untuk mendapatkan informasi detail saat memecahkan masalah, untuk sementara Anda dapat mengubah tingkat pencatatan `DEBUG`. Perubahan pada pengaturan catatan mulai berlaku saat Anda men-deploy grup. Untuk informasi selengkapnya, lihat the section called “Konfigurasi pencatatan untuk AWS IoT Greengrass”.

Pada sistem file lokal, AWS IoT Greengrass simpan catatan di lokasi berikut. Membaca catatan pada sistem file memerlukan izin root.

greengrass-root`/ggc/var/log/crash.log`

Menampilkan pesan yang dihasilkan ketika sebuah AWS IoT Greengrass core gagal.

greengrass-root`/ggc/var/log/system/runtime.log`

Menampilkan pesan tentang komponen mana yang gagal.

greengrass-root`/ggc/var/log/system/`

Berisi semua catatan dari AWS IoT Greengrass komponen sistem, seperti certificate manager dan connection manager. Dengan menggunakan pesan di `ggc/var/log/system/` dan `ggc/var/log/system/runtime.log`, Anda harus bisa mengetahui error mana yang terjadi di AWS IoT Greengrass komponen sistem.

greengrass-root`/ggc/var/log/system/localwatch/`

Berisi log untuk AWS IoT Greengrass komponen yang menangani pengunggahan log Greengrass ke Log. CloudWatch Jika Anda tidak dapat melihat log Greengrass, maka Anda dapat menggunakan log CloudWatch ini untuk pemecahan masalah.

greengrass-root`/ggc/var/log/user/`

Berisi semua catatan dari fungsi Lambda yang ditetapkan pengguna. Periksa folder ini untuk menemukan pesan error dari fungsi Lambda lokal Anda.

> **Note**
>
> Secara default, *greengrass-root* adalah `/greengrass` direktori. Jika menulis direktori dikonfigurasi, maka log berada di bawah direktori itu.

Jika log dikonfigurasi untuk disimpan di cloud, gunakan CloudWatch Log untuk melihat pesan log. `crash.log`hanya ditemukan di log sistem file pada perangkat AWS IoT Greengrass inti.

Jika AWS IoT dikonfigurasi untuk menulis log CloudWatch, periksa log tersebut jika kesalahan koneksi terjadi ketika komponen sistem mencoba untuk terhubungAWS IoT.

Untuk informasi lebih lanjut tentang AWS IoT Greengrass catatan, lihat the section called "Pemantauan dengan AWS IoT Greengrass log".

> **Note**
>
> catatan AWS IoT Greengrass perangkat lunak Core v1.0 disimpan di bawah *greengrass-root*`/var/log` direktori.

Pemecahan masalah penyimpanan

Ketika penyimpanan file lokal penuh, beberapa komponen mungkin mulai gagal:

- Pemutakhiran bayangan lokal tidak terjadi.
- Sertifikat server MQTT core baru AWS IoT Greengrass tidak dapat diunduh secara lokal.
- Deployment gagal.

Anda harus selalu mengetahui jumlah ruang kosong yang tersedia secara lokal. Anda dapat menghitung ruang kosong berdasarkan ukuran fungsi Lambda yang di-deploy konfigurasi pencatatan (lihat the section called "Pemecahan masalah dengan catatan"), dan jumlah bayangan yang disimpan secara lokal.

Pemecahan masalah pesan

Semua pesan dikirim secara lokal di AWS IoT Greengrass dikirim dengan QoS 0. Secara default, AWS IoT Greengrass menyimpan pesan dalam antrian in-memory. Oleh karena itu, pesan yang

belum diproses akan hilang ketika Greengrass core restart; sebagai contoh, setelah deployment grup atau perangkat reboot. Namun, Anda dapat mengonfigurasi AWS IoT Greengrass (v1.6 atau yang lebih baru) untuk pesan cache ke sistem file sehingga mereka bertahan di restart core. Anda juga dapat mengonfigurasi ukuran antrian. Jika Anda mengonfigurasi ukuran antrian, pastikan bahwa itu lebih besar dari atau sama dengan 262144 byte (256 KB). Jika tidak, AWS IoT Greengrass mungkin tidak dimulai dengan benar. Untuk informasi selengkapnya, lihat the section called “Antrean pesan MQTT”.

> **Note**
>
> Saat menggunakan antrian dalam memori default, kami sarankan Anda men-deploy grup atau restart perangkat ketika gangguan layanan terendah.

Anda juga dapat mengonfigurasi core untuk membuat sesi persisten dengan AWS IoT. Hal ini memungkinkan core untuk menerima pesan yang dikirim dari AWS Cloud sementara core offline. Untuk informasi selengkapnya, lihat the section called “Sesi persisten MQTT dengan AWS IoT Core”.

Memecahkan masalah timeout sinkronisasi bayangan

Penundaan signifikan dalam komunikasi antara perangkat core Greengrass dan cloud mungkin menyebabkan sinkronisasi bayangan gagal karena batas waktu. Dalam hal ini, Anda akan melihat entri log yang serupa dengan berikut ini:

```
[2017-07-20T10:01:58.006Z][ERROR]-cloud_shadow_client.go:57,Cloud shadow
 client error: unable to get cloud shadow what_the_thing_is_named for
 synchronization. Get https://1234567890abcd.iot.us-west-2.amazonaws.com:8443/things/
what_the_thing_is_named/shadow: net/http: request canceled (Client.Timeout exceeded
 while awaiting headers)
[2017-07-20T10:01:58.006Z][WARN]-sync_manager.go:263,Failed to get cloud
 copy: Get https://1234567890abcd.iot.us-west-2.amazonaws.com:8443/things/
what_the_thing_is_named/shadow: net/http: request canceled (Client.Timeout exceeded
 while awaiting headers)
[2017-07-20T10:01:58.006Z][ERROR]-sync_manager.go:375,Failed to execute sync operation
 {what_the_thing_is_named VersionDiscontinued []}"
```

Perbaikan yang mungkin adalah untuk mengonfigurasi jumlah waktu yang perangkat core menunggu respons host. Buka file config.json di *`greengrass-root`*`/config` dan tambahkan `system.shadowSyncTimeout` bidang dengan nilai timeout dalam detik. Sebagai contoh:

```
{
  "system": {
    "shadowSyncTimeout": 10
  },
  "coreThing": {
    "caPath": "root-ca.pem",
    "certPath": "cloud.pem.crt",
    "keyPath": "cloud.pem.key",
    ...
  },
  ...
}
```

Jika tidak `shadowSyncTimeout` nilai yang ditentukan dalam `config.json`, default adalah 5 detik.

> **Note**
>
> Untuk perangkat lunak Core AWS IoT Greengrass v1.6 dan sebelumnya, default-nya `shadowSyncTimeout` adalah 1 detik.

Periksa Re: AWS Posting

Jika Anda tidak dapat menyelesaikan masalah menggunakan informasi pemecahan masalah dalam topik ini, Anda dapat mencari Memecahkan masalah atau memeriksa AWS IoT Greengrasstag di AWS re:Post untuk masalah terkait atau memposting pertanyaan baru. Anggota AWS IoT Greengrass tim secara aktif memantau AWS re:post.

Riwayat dokumen untuk AWS IoT Greengrass

Tabel berikut menjelaskan perubahan penting pada Panduan AWS IoT Greengrass Pengembang setelah Juni 2018. Untuk notifikasi tentang pembaruan dokumentasi ini, Anda dapat berlangganan ke umpan RSS.

Perubahan	Deskripsi	Tanggal
Perbarui ke akhir dukungan untuk v1.11.x Snap	Memperbarui akhir informasi dukungan untuk AWS IoT Greengrass core v 1.11.x Snap di snapcraft.io.	September 22, 2023
Akhir dukungan untuk v1.11.x Snap	Menambahkan akhir informasi dukungan untuk AWS IoT Greengrass core v 1.11.x Snap di snapcraft.io.	September 19, 2023
Gambar Docker untuk v1.11.6 AWS IoT Greengrass	Gambar Docker untuk perangkat lunak AWS IoT Greengrass Core v1.11.6 tersedia di Amazon Elastic Container Registry (Amazon ECR) Registry ECR) dan Docker Hub. Kami merekomendasikan bahwa Anda selalu menjalankan versi terbaru.	12 April 2022
AWS IoT Device Tester (IDT) untuk penghentian AWS IoT Greengrass V1	IDT untuk AWS IoT Greengrass V1 tidak akan lagi menghasilkan laporan kualifikasi yang ditandatangani.	4 April 2022
Support update untuk AWS IoT Device Tester untuk AWS IoT Greengrass	IDT untuk AWS IoT Greengrass versi 4.4.1 sekarang mendukung penggunaan perangkat lunak AWS IoT	24 Maret 2022

	Greengrass inti versi v1.11.6 untuk kualifikasi perangkat.	
AWS IoT Greengrass versi 1.11.6 dirilis	Versi 1.11.6 dari perangkat lunak AWS IoT Greengras s Core tersedia. Versi ini berisi peningkatan performa dan perbaikan bug. Kami merekomendasikan bahwa Anda selalu menjalankan versi terbaru.	24 Maret 2022
SiteWise Konektor IoT versi 12 dirilis	Versi 12 dari SiteWise konektor IoT tersedia. Rilis ini berisi perbaikan bug.	Desember 23, 2021
Gambar Docker untuk AWS IoT Greengrass v1.11.5 dan v1.10.5	Gambar Docker untuk perangkat lunak AWS IoT Greengrass Core v1.11.5 dan v1.10.5 tersedia di Amazon Elastic Container Registry (Amazon ECR) Registry ECR) dan Docker Hub. Kami merekomendasikan bahwa Anda selalu menjalankan versi terbaru.	Desember 22, 2021
AWS IoT Greengrass V1 kebijakan pemeliharaan	Kebijakan AWS IoT Greengras s V1 pemeliharaan mendefini sikan berbagai tingkat pemeliharaan dan pembaruan untuk AWS IoT Greengrass V1 layanan dan perangkat lunak AWS IoT Greengrass inti v1.x.	Desember 20, 2021

AWS IoT Device Tester versi 4.4.1 dirilis	IDT untuk AWS IoT Greengrass versi 4.4.1 sekarang tersedia. Rilis ini mencakup suite AWS IoT Greengrass kualifikasi (GGQ) v1.3.1, dan mendukung penggunaan perangkat lunak AWS IoT Greengrass inti versi v1.11.5 dan v1.10.5 untuk kualifikasi perangkat.	Desember 20, 2021
AWS IoT Greengrass versi 1.11.5 dan 1.10.5 dirilis	Versi 1.11.5 dan 1.10.5 dari perangkat lunak AWS IoT Greengrass Core tersedia. Versi ini berisi peningkatan kinerja dan perbaikan bug. Kami merekomendasikan bahwa Anda selalu menjalankan versi terbaru.	Desember 12, 2021
Diterbitkan ulang gambar AWS IoT Greengrass v1.11.4 dan v1.10.4 Docker	Gambar Docker untuk perangkat lunak AWS IoT Greengrass Core versi 1.11.4 dan 1.10.4 telah diterbitkan ulang di Amazon Elastic Container Registry (Amazon ECR) Registry ECR) dan Docker Hub untuk mengatasi perbaikan bug. BusyBox Untuk menggunakan gambar Docker terbaru, gunakan `1.10.4-1` tag `1.11.4-1` atau. Untuk informasi selengkapnya tentang tag yang tersedia, lihat amazon/ aws-iot-greengrass di Docker Hub.	8 Desember 2021

CloudWatch Konektor metrik mendukung stempel waktu duplikat dalam data input	Anda sekarang dapat mengirim data input dengan stempel waktu duplikat ke konektor ini.	November 19, 2021
Pembaruan pencegahan wakil kebingungan lintas layanan	AWS IoT Greengrass mendukung penggunaan kunci konteks kondisi `aws:SourceAccount` global `aws:SourceArn` dan dalam kebijakan sumber daya IAM untuk mencegah masalah wakil yang membingungkan.	November 1, 2021
Gambar Docker untuk v1.11.4 AWS IoT Greengrass	Gambar Docker untuk perangkat lunak AWS IoT Greengrass Core v1.11.4 tersedia di Amazon Elastic Container Registry (Amazon ECR) Registry ECR) dan Docker Hub. Kami merekomendasikan bahwa Anda selalu menjalankan versi terbaru.	Agustus 24, 2021
Diterbitkan AWS IoT Greengrass v1.11.4 snap	Versi 1.11.4 dari AWS IoT Greengrass snap tersedia. Kami merekomendasikan bahwa Anda selalu menjalankan versi terbaru.	Agustus 20, 2021
Support update untuk AWS IoT Device Tester untuk AWS IoT Greengrass	IDT untuk AWS IoT Greengrass versi 4.1.0 sekarang mendukung penggunaan perangkat lunak AWS IoT Greengrass inti versi v1.11.4 untuk kualifikasi perangkat.	18 Agustus 2021

AWS IoT Greengrass versi 1.11.4 dirilis	Versi 1.11.4 dari perangkat lunak AWS IoT Greengras s Core tersedia. Rilis ini memperbaiki masalah dengan pengelola aliran yang mencegah peningkatan ke v1.11.3 dari versi perangkat lunak Core yang lebih lama. AWS IoT Greengrass Kami merekomendasikan bahwa Anda selalu menjalankan versi terbaru.	17 Agustus 2021
Titik akhir VPC () AWS PrivateLink	AWS IoT Greengrass sekarang mendukung antarmuka VPC endpoint (AWS PrivateLink) untuk bidang kontrol. AWS IoT Greengrass Anda dapat membuat koneksi pribadi antara VPC Anda dan pesawat AWS IoT Greengrass kontrol.	16 Agustus 2021
AWS IoT Device Tester versi 4.1.0 dirilis	Versi 4.1.0 dari AWS IoT Device Tester untuk AWS IoT Greengrass tersedia. Versi ini mendukung penggunaa n perangkat lunak AWS IoT Greengrass inti versi 1.11.3 dan 1.10.4 untuk kualifikasi perangkat.	23 Juni 2021

Diterbitkan AWS IoT Greengrass v1.11.3 snap	Versi 1.11.3 dari AWS IoT Greengrass snap berisi peningkatan kinerja dan perbaikan bug. Kami merekomendasikan bahwa sebagai praktik terbaik Anda selalu menjalankan versi terbaru.	15 Juni 2021
Gambar Docker untuk AWS IoT Greengrass v1.11.3 dan v1.10.4 dirilis	Gambar Docker untuk perangkat lunak AWS IoT Greengrass Core v1.11.3 dan v1.10.4 tersedia di Amazon Elastic Container Registry (Amazon ECR) Registry ECR) dan Docker Hub. Versi AWS IoT Greengrass Core ini berisi peningkatan kinerja dan perbaikan bug. Kami merekomendasikan bahwa Anda selalu menjalankan versi terbaru.	15 Juni 2021
AWS IoT Greengrass versi 1.11.3 dirilis	Versi 1.11.3 dari perangkat lunak AWS IoT Greengras s Core tersedia. Versi ini berisi peningkatan performa dan perbaikan bug. Kami merekomendasikan bahwa Anda selalu menjalankan versi terbaru.	14 Juni 2021

AWS IoT Greengrass versi 1.10.4 dirilis	Versi 1.10.4 dari perangkat lunak AWS IoT Greengras s Core tersedia. Versi ini berisi peningkatan performa dan perbaikan bug. Kami merekomendasikan bahwa Anda selalu menjalankan versi terbaru.	14 Juni 2021
Adaptor Protokol Modbus-TCP versi 2 dirilis	Versi 2 dari konektor Modbus-TCP Protocol Adapter tersedia. Rilis ini menambahk an dukungan untuk string sumber yang dikodekan ASCII, UTF8, dan ISO8859.	24 Mei 2021
Konektor penerapan aplikasi Docker versi 7 dirilis	Versi 7 dari konektor deployment aplikasi Greengrass Docker tersedia.	5 April 2021
AWS IoT Greengrass versi 1.11.1 dirilis	Versi 1.11.1 dari perangkat lunak AWS IoT Greengras s Core tersedia. Versi ini berisi peningkatan performa dan perbaikan bug. Kami merekomendasikan bahwa Anda selalu menjalankan versi terbaru.	29 Maret 2021

AWS IoT Device Tester versi 4.0.2 dirilis	Versi 4.0.2 dari AWS IoT Device Tester untuk AWS IoT Greengrass tersedia. Versi ini menggantikan IDT v4.0.0 dan menambahkan dukungan untuk perangkat lunak Core versi 1.11.1. AWS IoT Greengrass Ini juga memperbaiki masalah yang menyebabkan IDT untuk menutupi kesalahan Hardware Security Integration (HSI).	29 Maret 2021
SiteWise Konektor IoT versi 11 dirilis	Versi 11 dari SiteWise konektor IoT tersedia. Ini meluncurkan dukungan untuk string yang berisi karakter tersembunyi atau unprintable. Rilis ini juga mencakup peningkatan performa umum dan perbaikan bug.	24 Maret 2021
Dipublikasikan ulang AWS IoT Greengrass v1.11.0 snap	AWS IoT Greengrass snap versi 1.11.0 telah diterbitkan ulang di Snapcraft untuk mengatasi perbaikan bug dan kemungkinan crash aplikasi saat menggunakan penerjemah Python. AWS IoT Greengrass tidak menyediakan snaps untuk perangkat lunak versi 1.10 dan 1.9.	19 Maret 2021

Support update untuk AWS IoT Device Tester untuk AWS IoT Greengrass	IDT untuk AWS IoT Greengrass versi 4.0.0 sekarang mendukung penggunaan perangkat lunak AWS IoT Greengrass inti versi v1.10.3 untuk kualifikasi perangkat.	18 Maret 2021
Dipublikasikan ulang AWS IoT Greengrass v1.8.4 snap	AWS IoT Greengrass snap versi 1.8.4 telah diterbitkan ulang di Snapcraft untuk mengatasi perbaikan bug dan kemungkinan crash aplikasi saat menggunakan penerjemah Python.	15 Maret 2021
Gambar Docker AWS IoT Greengrass v1.11.0 yang diterbitkan ulang untuk ARMv7L	Gambar Docker untuk perangkat lunak AWS IoT Greengrass Core versi 1.11.0 untuk platform ARMv7L telah diterbitkan ulang di Amazon Elastic Container Registry (Amazon ECR) Registry ECR) dan Docker Hub untuk mengatasi perbaikan bug dan kemungkinan crash aplikasi saat menggunakan interpreter Python.	8 Maret 2021
AWS IoT Greengrass v1.10.3 Gambar Docker dirilis	Gambar Docker untuk perangkat lunak AWS IoT Greengrass Core versi 1.10.3 sekarang tersedia di Amazon Elastic Container Registry (Amazon ECR) Registry ECR) dan Docker Hub.	8 Maret 2021

Diterbitkan ulang gambar AWS IoT Greengrass v1.11.0 dan v1.9.4 Docker	Gambar Docker untuk perangkat lunak AWS IoT Greengrass Core versi 1.11.0 dan 1.9.4 telah diterbitkan ulang di Amazon Elastic Container Registry (Amazon ECR) Registry (Amazon ECR) dan Docker Hub untuk mengatasi perbaikan bug dan kemungkinan crash aplikasi saat menggunakan interpret er Python. Citra Docker untuk Armv7l belum diterbitkan ulang saat ini.	26 Februari 2021
AWS IoT Greengrass versi 1.10.3 dirilis	Versi 1.10.3 dari perangkat lunak AWS IoT Greengras s Core tersedia. Versi ini menambahkan properti konfigurasi core `systemCom ponentAuthTimeout` dan berisi perbaikan performa dan perbaikan bug. Kami merekomendasikan bahwa Anda selalu menjalankan versi terbaru.	24 Februari 2021

SiteWise Konektor IoT versi 10 dirilis	Versi 10 dari SiteWise konektor IoT tersedia. Rilis ini menyelesaikan masalah stabilitas dengan StreamManager klien saat koneksi terputus, dan meningkatkan penanganan nilai OPC-UA saat tidak ada. `SourceTimestamp` Gunakan SiteWise konektor IoT untuk mengirim data perangkat dan peralatan lokal ke properti aset di IoT SiteWise	22 Januari 2021
SiteWise Konektor IoT versi 9 dirilis	Versi 9 dari SiteWise konektor IoT tersedia. Ini meluncurkan dukungan untuk Greengrass kustom `StreamManager` tujuan stream, deadbanding OPC-UA, mode pemindaian khusus dan tingkat pemindaian kustom. Ini juga mencakup peningkatan kinerja selama pembaruan konfigurasi yang dibuat dari gateway IoT SiteWise . Gunakan SiteWise konektor IoT untuk mengirim data perangkat dan peralatan lokal ke properti aset di IoT SiteWise	15 Desember 2020

AWS IoT Device Tester versi 4.0.0 dirilis	Versi 4.0.0 dari AWS IoT Device Tester untuk AWS IoT Greengrass tersedia. Versi ini memungkinkan Anda untuk menggunakan IDT untuk mengembangkan dan menjalankan serangkai an tes kustom Anda untuk validasi perangkat. Ini juga mencakup aplikasi IDT kode yang ditandatangani untuk macOS dan Windows.	15 Desember 2020
AWS IoT Greengrass jepret v1.11	Versi 1.11.0 dari AWS IoT Greengrass snap mendukung fungsi Lambda noncontai nerized. Kami merekomen dasikan bahwa sebagai praktik terbaik Anda selalu menjalank an versi terbaru.	6 Desember 2020
SiteWise Konektor IoT versi 8 dirilis	Versi 8 dari SiteWise konektor IoT tersedia. Rilis ini meningkatkan stabilitas ketika konektor mengalami konektivitas jaringan intermite n. Gunakan SiteWise konektor IoT untuk mengirim data perangkat dan peralatan lokal ke properti aset di IoT. SiteWise	19 November 2020
Konektor Firehose Kinesis mendukung Tidak ada mode kontainer	Anda dapat menggunakan parameter `IsolationMode` untuk mengonfigurasi mode kontainerisasi untuk konektor.	19 Oktober 2023

Konektor penerapan aplikasi Docker versi 6 dirilis	Versi 6 dari konektor deployment aplikasi Greengrass Docker tersedia.	18 September 2020
AWS IoT Greengrass versi 1.11.0 dirilis	Versi 1.11.0 dari perangkat lunak AWS IoT Greengras s Core tersedia. Versi ini menambahkan fitur telemetri kondisi sistem dan pemeriksa an kondisi lokal API. Stream manager sekarang dapat mengekspor data ke Amazon Simple Storage Service (Amazon S3) dan IoT. SiteWise Versi ini juga berisi peningkatan performa dan perbaikan bug. Kami merekomendasikan bahwa Anda selalu menjalankan versi terbaru.	16 September 2020
SiteWise Konektor IoT versi 7 dirilis	Versi 7 dari SiteWise konektor IoT tersedia. Rilis ini memperbaiki masalah dengan metrik gateway. Gunakan SiteWise konektor IoT untuk mengirim data perangkat dan peralatan lokal ke properti aset di IoT. SiteWise	14 Agustus 2020
ServiceNow MetricBase Konektor Integrasi, Integrasi Splunk, dan Twilio Notificat ions mendukung Tidak ada mode kontainer	Anda dapat menggunakan parameter `IsolationMode` untuk mengonfigurasi mode kontainerisasi untuk konektor.	30 Juli 2020

Konektor SNS mendukung Tidak ada mode kontainer	Anda dapat menggunakan parameter `IsolationMode` untuk mengonfigurasi mode kontainerisasi untuk konektor.	6 Juli 2020
CloudWatch Konektor metrik mendukung Tidak ada mode kontainer	Anda dapat menggunakan parameter `IsolationMode` untuk mengonfigurasi mode kontainerisasi untuk konektor.	17 Juni 2020
AWS IoT Greengrass versi 1.10.2 dirilis	Versi 1.10.2 dari perangkat lunak AWS IoT Greengras s Core tersedia. Versi ini menambahkan properti konfigurasi core `mqttOpera tionTimeout` dan berisi perbaikan performa dan perbaikan bug. Kami merekomendasikan bahwa Anda selalu menjalankan versi terbaru.	8 Juni 2020
Pemasang pembelajaran mesin Tensorflow tidak digunakan lagi	AWS IoT Greengrass Installer machine learning yang sudah dikemas Tensorflow telah tidak digunakan lagi. Sampel machine learning telah diperbarui ke Python 3.7.	29 Mei 2020
Dukungan kerangka kerja Chainer dan penginsta l pembelajaran mesin Greengrass tidak digunakan lagi	AWS IoT Greengrass Installer machine learning yang sudah dikemas dan download untuk MXNet dan DLR sudah tidak digunakan lagi. Support kerangka kerja chainer dan unduhan terkait tidak lagi digunakan.	4 Mei 2020

SiteWise Konektor IoT versi 6 dirilis	Versi 6 dari SiteWise konektor IoT tersedia. Rilis ini menambahkan dukungan untuk CloudWatch metrik dan penemuan otomatis tag OPC-UA baru. Ini berarti Anda tidak perlu memulai ulang gateway ketika tag berubah untuk sumber OPC-UA Anda. Versi konektor ini memerlukan manajer aliran dan perangkat lunak AWS IoT Greengras s Core v1.10.0 atau lebih tinggi. Gunakan SiteWise konektor IoT untuk mengirim data perangkat dan peralatan lokal ke properti aset di IoT. SiteWise	29 April 2020
Konektor ditingkatkan ke Python 3.7	Konektor yang mendukung waktu aktif Python telah diperbarui ke Python 3.7. Kami merekomendasikan Anda memperbarui versi konektor Anda dari Python 2.7 ke Python 3.7.	29 April 2020
Pengaturan perangkat Greengrass dapat berjalan dalam mode senyap	Anda dapat menjalank an pengaturan perangkat Greengrass dalam modus diam sehingga skrip tidak meminta Anda mengenai nilai-nilai apa pun.	27 April 2020

Gambar dasar Docker baru	Anda dapat mengunduh gambar AWS IoT Greengrass Docker yang dibangun di atas gambar dasar Alpine Linux (x86_64, ARMv7L, atau AArch64).	23 April 2020
AWS IoT Greengrass versi 1.10.1 dirilis	Versi 1.10.1 dari perangkat lunak AWS IoT Greengrass Core tersedia. Versi ini berisi peningkatan performa dan perbaikan bug. Kami merekomendasikan bahwa Anda selalu menjalankan versi terbaru.	16 April 2020
Bab keamanan baru	AWS IoT Greengrass konten keamanan telah ditata ulang, dengan informasi baru ditambahkan.	30 Maret 2020
Gunakan manajer paket APT untuk menginstal AWS IoT Greengrass	Pada distribusi Linux berbasis Debian yang didukung, Anda dapat menggunakan `apt` untuk menginstal perangkat lunak AWS IoT Greengrass Core pada perangkat Anda.	26 Februari 2020

SiteWise Konektor IoT versi 5 dirilis	Versi 5 dari SiteWise konektor IoT tersedia. Rilis ini memperbaiki masalah kompatibilitas dengan perangkat lunak AWS IoT Greengrass Core v1.9.4. Gunakan SiteWise konektor IoT untuk mengirim data perangkat dan peralatan lokal ke properti aset di IoT. SiteWise	12 Februari 2020
Skrip baru untuk mengatur perangkat inti dengan cepat	Anda dapat menggunak an pengaturan perangkat Greengrass untuk mengonfig urasi perangkat core Anda dalam hitungan menit. Juga, AWS IoT Greengrass sekarang mendukung fungsi Node.js 12.x Lambda.	20 Desember 2019
AWS IoT Greengrass versi 1.10.0 dirilis	Versi 1.10.0 dari perangkat lunak AWS IoT Greengrass Core tersedia. Fitur-fitur baru di versi ini termasuk pengelola Aliran, dukungan kontainer dengan konektor deploymen t aplikasi Docker, fungsi Lambda nonkontainerisasi yang dapat mengakses sumber daya machine learning, dukungan untuk sesi persisten MQTT dengan AWS IoT, dan dukungan untuk lalu lintas MQTT lokal melalui port tertentu.	25 November 2019

Dukungan konsol untuk pemberitahuan penerapan	Gunakan EventBridge konsol Amazon untuk membuat aturan peristiwa yang memicu saat penerapan grup Greengrass Anda mengubah status.	14 November 2019
AWS IoT Greengrass versi 1.9.4 dirilis	Versi 1.9.4 dari perangkat lunak AWS IoT Greengras s Core tersedia. Versi ini berisi peningkatan performa dan perbaikan bug. Sebagai praktik terbaik, kami merekomendasikan bahwa Anda selalu menjalankan versi terbaru.	17 Oktober 2019
Dukungan konsol untuk mengelola peran layanan Greengrass	Gunakan fitur baru dan yang lebih baik di AWS IoT konsol untuk mengelola peran layanan Greengrass Anda.	4 Oktober 2019
Dukungan konsol untuk mengelola tag tingkat grup	Anda dapat membuat, melihat, dan mengelola tanda untuk grup Greengrass Anda di konsol AWS IoT tersebut.	23 September 2019
Konektor pembelajaran mesin baru	Gunakan konektor ML Feedback untuk menerbitkan model input dan prediksi dan konektor ML Object Detection untuk menjalankan layanan inferensi deteksi objek lokal.	19 September 2019

AWS IoT Greengrass versi 1.9.3 dirilis	Versi 1.9.3 dari perangkat lunak AWS IoT Greengrass Core tersedia. Versi ini memungkinkan Anda untuk menginstal perangkat lunak AWS IoT Greengrass Core pada distribusi Raspbian pada arsitektur ARMv6L, mendukung pembaruan OTA pada port 443 dengan ALPN, dan berisi perbaikan bug untuk muatan biner yang dikirim dari fungsi Lambda Python 2.7 ke fungsi Lambda lainnya.	12 September 2019
AWS IoT Greengrass versi 1.8.4 dirilis	Versi 1.8.4 dari perangkat lunak AWS IoT Greengrass Core tersedia. Versi ini berisi peningkatan performa dan perbaikan bug. Jika Anda menjalankan v1.8.x, kami rekomendasikan agar Anda memperbarui ke v1.8.4 atau v1.9.3. Untuk versi sebelumnya, kami rekomendasikan Anda memperbarui ke v1.9.3.	30 Agustus 2019
AWS IoT Greengrass versi 1.9.2 dirilis dengan dukungan untuk OpenWrt	Versi 1.9.2 dari perangkat lunak AWS IoT Greengrass Core tersedia. Versi ini memungkinkan Anda untuk menginstal perangkat lunak AWS IoT Greengrass Core pada OpenWrt distribusi dengan arsitektur Armv8 (AArch64) dan ARMv7L.	20 Juni 2019

AWS IoT Greengrass versi 1.8.3 dirilis	Versi 1.8.3 dari perangkat lunak AWS IoT Greengrass Core tersedia. Versi ini berisi peningkatan performa umum dan perbaikan bug. Jika Anda menjalankan v1.8.x, kami rekomendasikan agar Anda memperbarui ke v1.8.3 atau v1.9.2. Untuk versi sebelumnya, kami rekomendasikan Anda memperbarui ke v1.9.2.	20 Juni 2019
AWS IoT Greengrass versi 1.9.1 dirilis	Versi 1.9.1 dari perangkat lunak AWS IoT Greengrass Core tersedia. Versi ini berisi perbaikan bug untuk pesan dari AWS IoT yang berisi karakter wildcard dalam topik.	10 Mei 2019
AWS IoT Greengrass versi 1.8.2 dirilis	Versi 1.8.2 dari perangkat lunak AWS IoT Greengrass Core tersedia. Versi ini berisi peningkatan performa umum dan perbaikan bug. Jika Anda menjalankan v1.8.x, kami rekomendasikan agar Anda memperbarui ke v1.8.2 atau v1.9.0. Untuk versi sebelumnya, kami rekomendasikan Anda memperbarui ke v1.9.0.	2 Mei 2019
AWS IoT Greengrass versi 1.9.0 dirilis	Fitur baru: Support untuk waktu aktif Lambda Python 3.7 dan Node.js 8.10, koneksi MQTT yang dioptimalkan, dan dukungan kunci Elliptic Curve (EC) untuk server MQTT lokal.	1 Mei 2019

AWS IoT Greengrass versi 1.8.1 dirilis	Versi 1.8.1 dari perangkat lunak AWS IoT Greengras s Core tersedia. Versi ini berisi peningkatan performa umum dan perbaikan bug. Sebagai praktik terbaik, kami merekomendasikan bahwa Anda selalu menjalankan versi terbaru.	18 April 2019
AWS IoT Greengrass snap tersedia di snapcraft	Gunakan aplikasi AWS IoT Greengrass Snap Store untuk merancang, menguji, dan menyebarkan perangkat lunak dengan cepat di perangkat Linux. AWS IoT Greengrass	1 April 2019
Support untuk kontrol akses lebih lanjut menggunakan izin berbasis tag	Anda dapat menggunak an kebijakan tag dalam AWS Identity and Access Management (IAM) untuk mengontrol akses ke AWS IoT Greengrass sumber daya Anda.	29 Maret 2019
Konektor IoT Analytics dirilis	Gunakan konektor IoT Analytics untuk mengirim data perangkat lokal ke AWS IoT Analytics saluran.	15 Maret 2019
Dukungan Batch pada konektor Firehose Kinesis	Konektor Kinesis Firehose mendukung pengiriman catatan data batch ke Amazon Data Firehose pada interval tertentu.	15 Maret 2019

AWS CloudFormation dukungan untuk AWS IoT Greengrass sumber daya	Gunakan AWS CloudFormation template untuk membuat dan mengelola AWS IoT Greengrass sumber daya.	15 Maret 2019
AWS IoT Greengrass versi 1.8.0 dirilis	Fitur baru: Identitas akses default yang dapat dikonfigurasi untuk fungsi Lambda, dukungan untuk lalu lintas HTTPS melalui port 443, dan ID klien bernama yang dapat diprediksi untuk koneksi MQTT dengan. AWS IoT	7 Maret 2019
AWS IoT Greengrass versi 1.7.1 dan 1.6.1 dirilis	Versi 1.7.1 dan 1.6.1 dari perangkat lunak AWS IoT Greengrass Core tersedia. Versi ini memerlukan kernel Linux versi 3.17 atau yang lebih baru. Kami merekomendasikan bahwa pelanggan yang menjalankan versi apa pun untuk segera memperbarui perangkat lunak core Greengrass ke versi 1.7.1.	11 Februari 2019
SageMaker Runtime pembelajaran mendalam Neo	Runtime pembelajaran mendalam SageMaker Neo mendukung model pembelajaran mesin yang telah dioptimalkan oleh kompiler pembelajaran mendalam SageMaker Neo.	28 November 2018

Jalankan AWS IoT Greengrass dalam wadah Docker	Anda dapat menjalankan AWS IoT Greengrass dalam wadah Docker dengan mengonfigurasi grup Greengrass Anda agar berjalan tanpa kontainerisasi.	26 November 2018
AWS IoT Greengrass versi 1.7.0 dirilis	Fitur baru: Konektor Greengrass, secrets manager lokal, pengaturan isolasi dan izin untuk fungsi Lambda, root perangkat keras dari keamanan kepercayaan, koneksi menggunakan ALPN atau proxy jaringan, dan dukungan Raspbian Stretch.	26 November 2018
AWS IoT Greengrass unduhan perangkat lunak	Paket perangkat lunak AWS IoT Greengrass Core, AWS IoT Greengrass Core SDK, dan AWS IoT Greengrass Machine Learning SDK tersedia untuk dowload melalui Amazon. CloudFront	26 November 2018
AWS IoT Device Tester untuk AWS IoT Greengrass	Gunakan AWS IoT Device Tester AWS IoT Greengrass untuk memverifikasi bahwa arsitektur CPU, konfigurasi kernel, dan driver Anda berfungsi. AWS IoT Greengrass	26 November 2018

AWS CloudTrail logging untuk panggilan AWS IoT Greengrass API	AWS IoT Greengrass terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di AWS IoT Greengrass.	29 Oktober 2018
Dukungan untuk TensorFlow v1.10.1 di NVIDIA Jetson TX2	Pustaka yang TensorFlow dikompilasi sebelumnya untuk NVIDIA Jetson TX2 yang AWS IoT Greengrass menyediakan sekarang menggunakan v1.10.1. TensorFlow Ini mendukung Jetpack 3.3 dan CUDA Toolkit 9.0.	18 Oktober 2018
Support untuk sumber daya machine learning MxNet v1.2.1	AWS IoT Greengrass mendukung model pembelajaran mesin yang dilatih menggunakan MXNet v1.2.1.	29 Agustus 2018
AWS IoT Greengrass versi 1.6.0 dirilis	Fitur baru: Lambda executable, antrean pesan dikonfigurasi, dikonfigurasi kembali interval coba lagi, sumber daya volume di bawah /proc, dan direktori tulis dikonfigurasi.	26 Juli 2018

Pembaruan sebelumnya

Tabel berikut menjelaskan perubahan penting pada Panduan AWS IoT Greengrass Pengembang sebelum Juli 2018.

Perubahan	Deskripsi	Tanggal
AWS IoT Greengrass Versi 1.5.0 Dirilis	Fitur-fitur baru: • Inferensi machine learning lokal menggunakan model cloud-trained. Untuk informasi selengkapnya, lihat Lakukan inferensi machine learning. • Fungsi Lambda Greengrass mendukung data input biner, selain JSON. Untuk informasi lebih lanjut, lihat AWS IoT Greengrass versi Core.	29 Maret 2018
AWS IoT Greengrass Versi 1.3.0 Dirilis	Fitur-fitur baru: • Agen pembaruan O ver-the-air (OTA) yang mampu menangani pekerjaan pembaruan Greengrass yang diterapkan di cloud. Untuk informasi selengkapnya, lihat Perbarui OTA AWS IoT Greengrass perangkat lunak Core. • Mengakses periferal lokal dan sumber daya dari fungsi Lambda Greengrass. Untuk informasi selengkapnya, lihat Akses sumber daya lokal dengan fungsi dan konektor Lambda.	27 November 2017
AWS IoT Greengrass Versi 1.1.0 Dirilis	Fitur-fitur baru: • Setel ulang AWS IoT Greengrass grup yang digunakan. Untuk informasi selengkapnya, lihat Atur ulang deployment. • Support untuk waktu aktif Lambda Node.js 6.10 dan Java 8, selain Python 2.7.	20 September 2017
AWS IoT Greengrass Versi 1.0.0 Dirilis	AWS IoT Greengrass umumnya tersedia.	7 Juni 2017